High voltage command board (CREAM experiment)

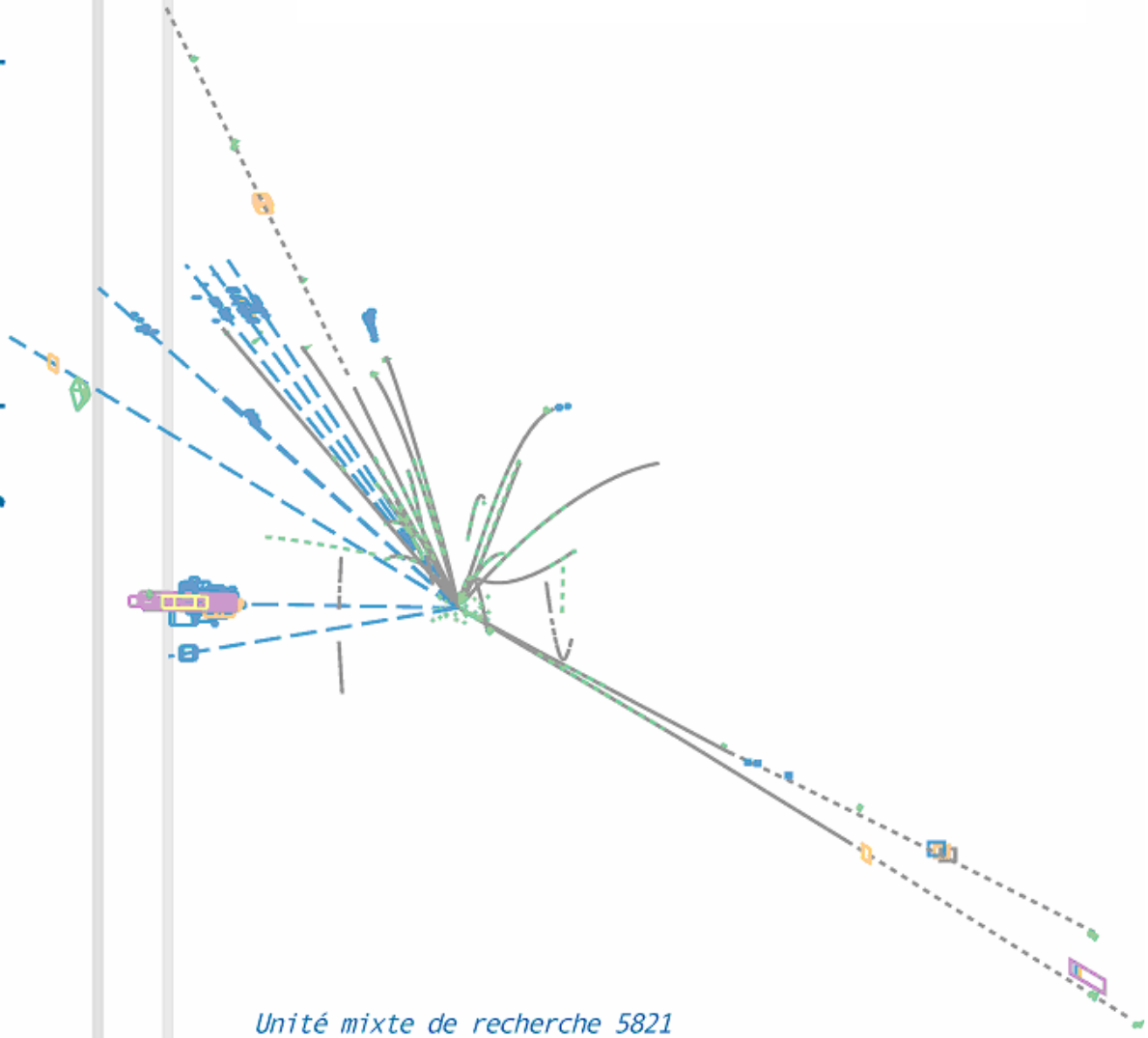
J. Bouvier, Ousmane Traore

Internal report LPSC 07-103
Version 3
August 2007

High voltage command board (CREAM experiment)

Board which realize the interface between the CREAM experiment primary computer unit and the High Voltage generator units (2 x 50 units) located near the PMT matrix.

Joël BOUVIER
Olivier TRAORE
Data Acquisition Team



Laboratoire de Physique Subatomique et de Cosmologie

Unité mixte de recherche 5821
CNRS-IN2P3 / UJF / INPG

1. Overview

The goal of this board is to realize an interface between the CREAM primary computer unit and the High Voltage generator units (2 x 50 units) located near the PMT matrix.
The CREAM experiment need two boards because the High Voltage generator units are located into 2 places on opposite side of the PMT matrix (one meter spaced).

The High voltage generators are voltage controlled generator and needs, for each of them:

- ✓ A validation bit, logical information where logical '0' is inhibition and logical '1' is validation
- ✓ An analog voltage which determine the output voltage value.

The link with the primary computer unit is done by a serial link described below.

This board is also useful like interfaces between the housekeeping board and the high voltage modules :

- ✓ The high voltage modules are powered through the board from the CHERCAM power supply to the high voltage modules.
- ✓ Th high voltage housekeeping information go through the board from the high voltage modules to the housekeeping board

2. High voltage generators characteristics

Here is a summary characteristics of the high voltage generator, for more information see the High Voltage generators data sheet.

o Logical module characteristics

The input impedance is equal to 1 MΩ.
The absolute maximum input voltage is equal to +/- 20V.
A logical low level is defined by a voltage which is less than 1,9V and a logical high level by a voltage greater then 2,2V.

o Analog module characteristics

The input impedance is equal to 100 KΩ.
The output voltage relation is five volts in command generate 1000V in output with an output absolute maximum value of 1200 V.

3. In command serial link description

The command is sending by the primary computer unit by a serial link. This link uses an RS422 electrical standard and the validate command are described below.
Validate commands can arrive anytime on the two modules.

o CHERCAM high voltage High level protocol

The typical command frame seen by the high voltage module is defined as below:

Device Address
DATA3
DATA2
DATA1
DATA0

There will be 100 high voltage modules (50 for each board); each of them will have an individual device address. Each module manage a cluster of 16 PM.

- **High Voltage level setting command**

  This command is used for setting the high voltage level of each high voltage module.

Device Address = (from 0x01 to 0x64)
DATA3 = 0xAA
DATA2 = DAC_value MSB
DATA1 = DAC_value LSB
DATA0 = unused

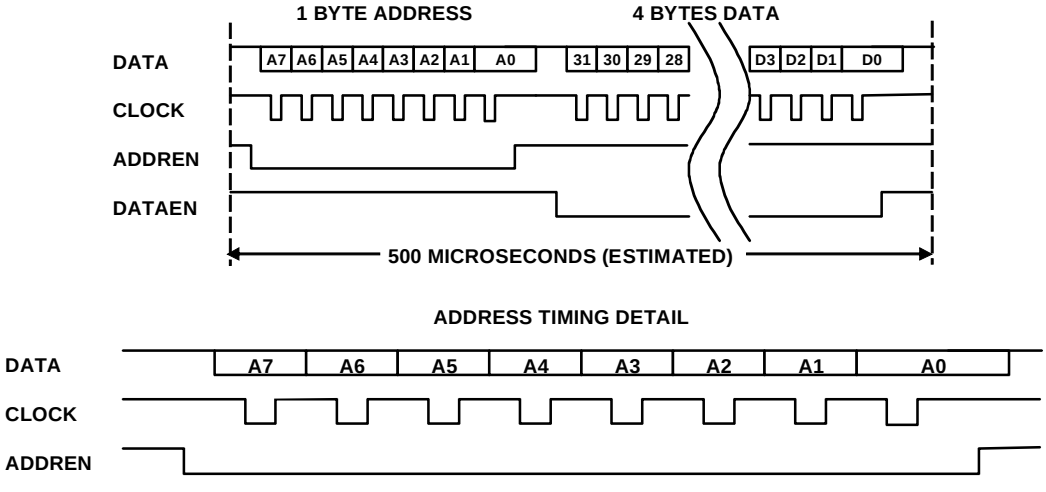
- **High Voltage disable/enable setting command**

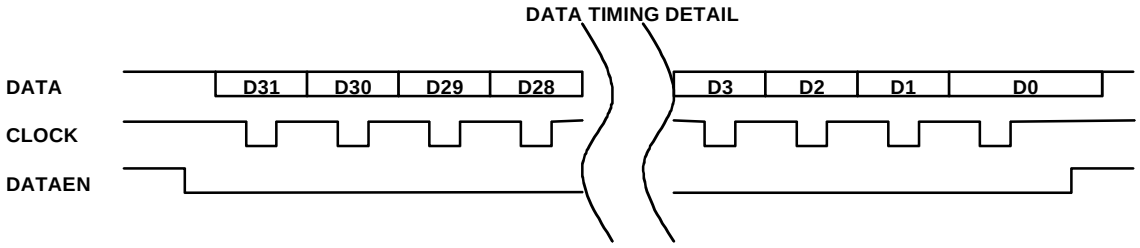
  This command is used to enable/disable the high voltage output of each high voltage module.

Device Address = (from 0x01 to 0x64)
DATA3 = (0x0A for enable, 0x05 for disable)
DATA2 = unused
DATA1 = unused
DATA0 = unused

o **Serial link timing**

The timings are shown below, the data is sampled on the rising edge of the clock.





The RESET signal (not shown in the previous timing diagrams) is a low level active signal.

4. DAC Board design

The board design is a FPGA based design. This solution has been adopted instead of basic logic component because the second solution need a lot of component, in particular for the generation of the VALIDx information, and the printed board has a surface constraint

All the software command are acknowledged by the FPGA which :

- ✓ generate all the signal command to the other board components (DAC components, Operational amplifier)
- ✓ put in a correct format the data from the computer unit to the DAC component
- ✓ memorize the VALID bit state for each HT generator
- ✓ check if the command data must be interpreted (in case of one channel link command the two DAC_BOARD and if this option is active by a correct mount of some on board resistor)

The design can be divided into 5 subsystems :

- ✓ input / output connector and level translator
- ✓ FPGA
- ✓ Power
- ✓ DACs
- ✓ Amplifier and line level adapter & driver

o Board synoptic

The Annex 1 shown a board synoptic where all the sub systems can be seen.

o input / output connector and level translator

All the information (data, power, command …) must be connected easily to the board to facilitate the board maintenance. It is the reason that all the input/Output is connected to the board with connectors.

▪ input power connector

The DAC_BOARD is supplied by the +12v delivered by a dedicated power source through a screwed connector (connector type PCB connector with screw clamp connection).
This power source was specially developed to supply all the subsystems of the CHERCAM detector.
The DAC_BOARD has 2 input power connector which has the same pinout. This one is given in the following figure.

SIGNAL	NAME	PIN
PMT supply power line	-3V3	1
PMT supply power line	+28V	2
Ground	Gnd	3
DAC board power line	+12V	4
Ground	Gnd	5
Ground	Gnd	6

Figure 1 : Power connector pin assignment

- **input serial link connector**

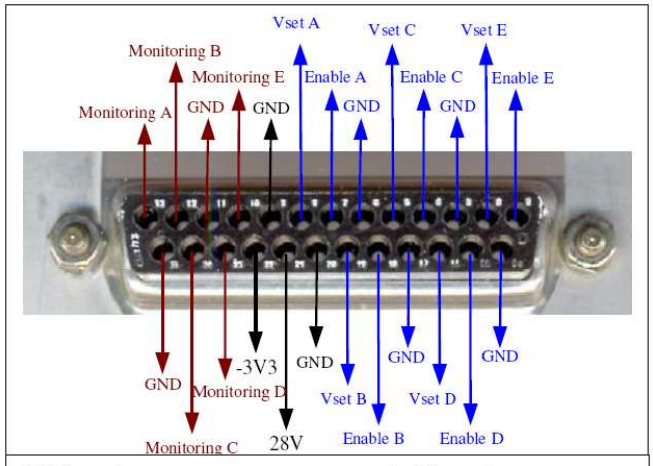
The signals which come from the computer unit are connected to the DAC_BOARD through a HE10-14 connector type.
The level signals are compatible with the RS422 specification.

SIGNAL	NAME	PIN (+)	PIN (-)
Address Enable	nADDREN	1	2
Data Enable	nDATAEN	3	4
Data	DATA	5	6
Clock	CLOCK	7	8
Reset	nRESET	9	10
Spare - NC		11	12
Ground	GROUND	13	14

Figure 2 : Serial link connector pin assignment

- **output connector to the HT module**

The signals generated to or coming from the PMT power module are connected to the DAC_BOARD through a SUBD-25 connector type.
The connector pinout is given in the following figures.



Câblage du connecteur par nappe sertie 25 conducteurs [1]

Figure 3 : Output connector to the HT module pin assignment (picture)

[1] "PINOUT DES MODULES HAUTE TENSION PAR 5" Ludovic ERAUD, Electronic team, LPSC, Grenoble

SIGNAL	NAME	PIN
Output validation for the channel 5	Enable E	1
Voltage setting for the channel 5	Vset E	2
Ground	Gnd	3
Output validation for the channel 3	Enable C	4
Voltage setting for the channel 3	Vset C	5
Ground	Gnd	6
Output validation for the channel 1	Enable A	7
Voltage setting for the channel 1	Vset A	8
Ground	Gnd	9
Voltage return for the channel 5	Monitoring E	10
Ground	Gnd	11
Voltage return for the channel 2	Monitoring B	12
Voltage return for the channel 1	Monitoring A	13
Ground	Gnd	14
Output validation for the channel 4	Enable D	15
Voltage setting for the channel 4	Vset D	16
Ground	Gnd	17
Output validation for the channel 2	Enable B	18
Voltage setting for the channel 2	Vset B	19
Ground	Gnd	20
Power	+28V	21
Power	-3V3	22
Voltage return for the channel 4	Monitoring D	23
Voltage return for the channel 3	Monitoring C	24
Ground	Gnd	25

Figure 4 : Output connector to the HT module pin assignment (text)

- **output connector to the HOUSEKEEPING board**

the link between the DAC_BOARD and the HOUSEKEEPING board is done with 2 ribbon cables.
There are connected to the boards through a HE10-50 vertical connector type.
The connector pinout is done in the following figure where i represents the connector number (i is included in the range 0 to 1).

SIGNAL	NAME	PIN	PIN	NAME	SIGNAL
Voltage return from the PMT HT power 0 + i	Mon<0+i>	1	2	Gnd	Ground
Voltage return from the PMT HT power 2 + i	Mon<2+i>	3	4	Gnd	Ground
Voltage return from the PMT HT power 4 + i	Mon<4+i>	5	6	Gnd	Ground
Voltage return from the PMT HT power 6 + i	Mon<6+i>	7	8	Gnd	Ground
Voltage return from the PMT HT power 8 + i	Mon<8+i>	9	10	Gnd	Ground
Voltage return from the PMT HT power 10 + i	Mon<10+i>	11	12	Gnd	Ground
Voltage return from the PMT HT power 12 + i	Mon<12+i>	13	14	Gnd	Ground
Voltage return from the PMT HT power 14 + i	Mon<14+i>	15	16	Gnd	Ground
Voltage return from the PMT HT power 16 + i	Mon<16+i>	17	18	Gnd	Ground
Voltage return from the PMT HT power 18 + i	Mon<18+i>	19	20	Gnd	Ground

SIGNAL	NAME	PIN	PIN	NAME	SIGNAL
Voltage return from the PMT HT power 20 + i	Mon<20+i>	21	22	Gnd	Ground
Voltage return from the PMT HT power 22 + i	Mon<22+i>	23	24	Gnd	Ground
Voltage return from the PMT HT power 24 + i	Mon<24+i>	25	26	Gnd	Ground
Voltage return from the PMT HT power 26 + i	Mon<26+i>	27	28	Gnd	Ground
Voltage return from the PMT HT power 28 + i	Mon<28+i>	29	30	Gnd	Ground
Voltage return from the PMT HT power 30 + i	Mon<30+i>	31	32	Gnd	Ground
Voltage return from the PMT HT power 32 + i	Mon<32+i>	33	34	Gnd	Ground
Voltage return from the PMT HT power 34 + i	Mon<34+i>	35	36	Gnd	Ground
Voltage return from the PMT HT power 36 + i	Mon<36+i>	37	38	Gnd	Ground
Voltage return from the PMT HT power 38 + i	Mon<38+i>	39	40	Gnd	Ground
Voltage return from the PMT HT power 40 + i	Mon<40+i>	41	42	Gnd	Ground
Voltage return from the PMT HT power 42 + i	Mon<42+i>	43	44	Gnd	Ground
Voltage return from the PMT HT power 44 + i	Mon<44+i>	45	46	Gnd	Ground
Voltage return from the PMT HT power 46 + i	Mon<46+i>	47	48	Gnd	Ground
Voltage return from the PMT HT power 48 + i	Mon<48+i>	49	50	Gnd	Ground

Figure 5 : output connector to the HOUSEKEEPING board (pinout)

- **Power**

The POWER subsystems generate all the necessary voltage used on the DAC_BOARD :

- 3V3 : used by the FPGA component, the DACs components and the interface components
- 2V5 : used by the FPGA component
- 9V : used by the voltage comparators (generation of the validation signal for the HT power module, the operational amplifiers (voltage setting for the HT power module)

The 3V3 and 2V5 are generated by 2 “SYNCHRONOUS STEP-DOWN CONVERTER » (one per output voltage).
The 9V is generate by a linear regulator from the DAC_BOARD input power (12V).
The others power line (28V and –3V3) are not used on the board and only send to the HT power module.
The following figure shown the consumption of each component and the consumption on each voltage

Component	Number	3,3V				2V5				9V			
		Unit		Total		Unit		Total		Unit		Total	
		typ	max	typ	max	typ	max	typ	max	typ	max	typ	max
APA300BG456-IND	1	31,14	31,14	31,14	31,14	8,57	8,57	8,57	8,57				
AM26LV32	2	9,00	15,00	18,00	30,00								
Ref. tension : REF3020	7	0,06	0,06	0,41	0,41								
AD5328	7	0,70	1,50	4,90	10,50								
Pont AOP + FANOUT	50									0,39	0,39	19,50	19,50
AOP : TS934	13									0,08	0,12	1,04	1,61
Pont comp + FANOUT	13									0,10	0,10	1,30	1,30
Comparator TLC3704	13									0,05	0,10	0,65	1,30
Total				54,45	72,05			8,57	8,57			22,49	**23,71**

Figure 6 : components power supply

The power of the board can be expressed as being the sum of the power absorptive by the linear regulator and that absorptive by the step-down dc-dc converters.
This power is equal to :

Board_power = linear_regulateur_power + step-down regulator
= 12 x I linear_reg + (I_{3V3} x 3,3 + I_{2V5} x 2,5) x (1 / Efficiency_step-down_reg)
= 12 x 23,71 + (72,05 x 3,3 + 8,57 x 2,5) x (1 / 0,9) in mW
= 284,52 + (237,765 + 21,75) x 1,11
= 284,52 + 288,06
Board_power = 572,58 mW (max)

with Efficiency_step-down_regulator equal to 90 %

- **DACs**

A followed paragraph (paragraph 6) explains how the Digital to Analog Converter works.
The goal of this subsystem is to convert digital data (12 bits) into analog signal in range 0 to 2,048 V.
The analog signal range is given by an external reference to the DAC chip (a SOT23-3 CMOS VOLTAGE REFERENCE chip).
The DACs output is set to 0 after a power on (DAC chip characteristics).

- **Amplifier and line level adapter & driver**

The goal of this subsystem is to convert :

- ✓ The HT valid signal coming from the FPGA to 0-9v valid signal to the HT module.
  This function is assumed by a comparator type component which have a comparison threshold set to 1/10 of his supply voltage.
  The threshold is make with 10K/100K network resistor.

- ✓ The voltage setting coming from the DACs component to 0-6v analog signal to the HT module.
  This function is assumed by an operational amplifier used in a non inverter amplifier, supplied by a 9v power supply and which have a gain of 3 set by a 10K/20K network resistor.

The following figure shows the functional block diagram of this sub system.

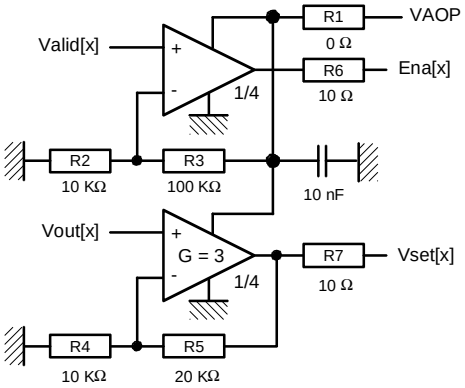


Figure 7 : Amplifier and line level adapter functional block diagram

- **FPGA**

The FPGA is the functional core of the DAC_BOARD.

It translate the stream from the CREAM primary computer to the HT modules in accordance with the characteristics ones of the serial input and components present on the board
To minimize the board dynamic consumption power, there are no clock generator on the board and the FPGA use the serial link input clock to set the parameter.
The FPGA pinout is given in Annex 3 : FPGA Pin report (list per name)and in Annex 4 : FPGA Pin report (list per pin).
The shows the signal generation from the FPGA to the DAC chip.

- **FGPA programmation connector**

The connector pin out and the connection between this one and the FPGA are given in the **Figure 8**.
All the interconnection are not necessary due of the FPGA type.
So the resistor R1 and R2 are not necessary because the FPGA are always supplied by the onboard DC/DC converter which generates all the voltage necessary for a good work of the board[3].
The capacitor (4,7 µF and 0,1 µF) are also not necessary because there is only one FPGA to be programmed on the board[3].
The signal RCK is not connected from the connector to the FPGA and as the FPGA pin has an internal pull up this pin is left floating[2].

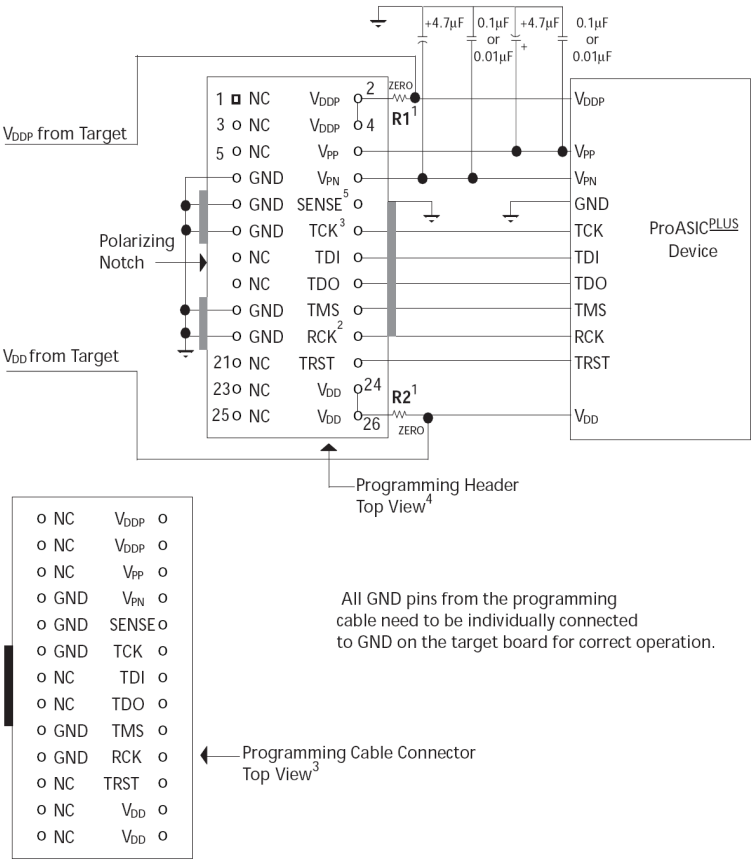


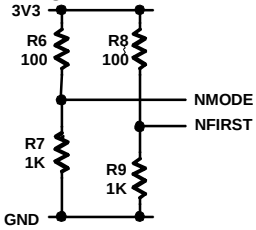
Figure 8: ProASIC PLUS ISP Board Layout and Programming Connector Top View[3]

5. Setting

The DAC_BOARD is made like a generic board capable to manage either the first 50 HT module or the last 50 one.

[2] ProASICPLUS Flash Family FPGAs, November 2004
[3] ACTEL : In-System Programming ProASICPLUS Devices (Application Note June 2004)

Several functioning mode are available for the board.
Each of them represents a functioning mode can simplified either the hardware interconnection or the software.
2 indicators set 3 functioning mode and they set with a network resistor like the following figure (the setting mode with discrete resistor is chosen to have a good resistance with the vibrations).

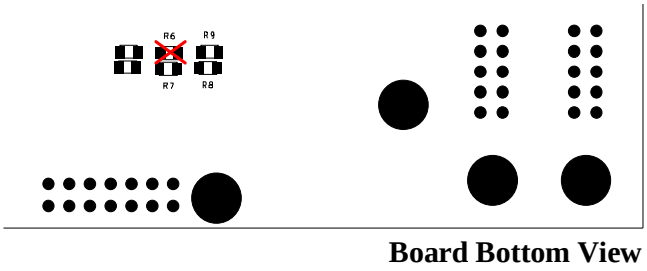


the first indicator, nMODE, set the DAC_BOARD to manage 50 HT module without test about the HT module number, this one is made by software.
When the nMODE bit is set to 0, the other indicator have no signification, it have one only if the nMODE bit is set to 1.

The second indicator, nFIRST, set the DAC_BOARD to manage either the first 50 HT module (nFIRST = 0) or the last 50 one (nFIRST = 1).

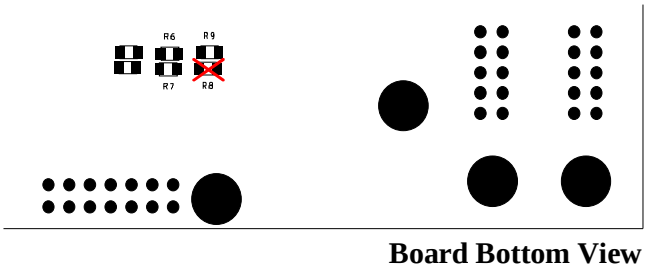
The following table resume the functioning mode.

if NMODE = 0 then 0 < @ < 50



if NMODE = 1 then 0 < @ < 100

with nFIRST = 0 => 0 < @ < 50 (default value)



nFIRST = 1 => 51 < @ < 100

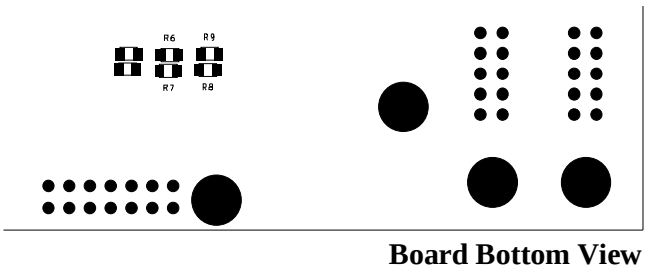


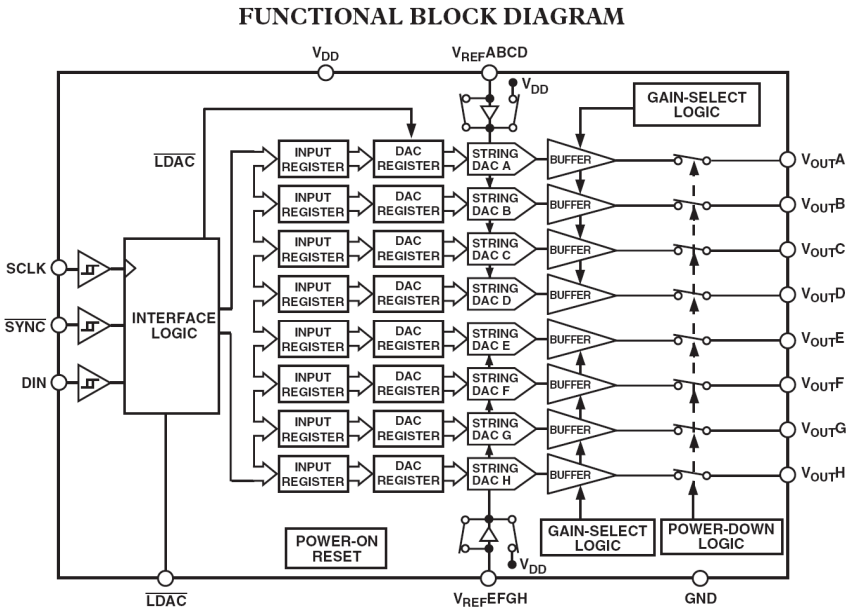
Figure 9 : DAC_BOARD functioning set mode

6. The Digital to Analog Converter : AD5328 (Analog Devices)

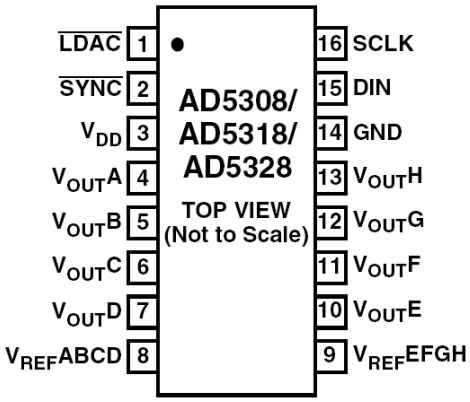
o General description

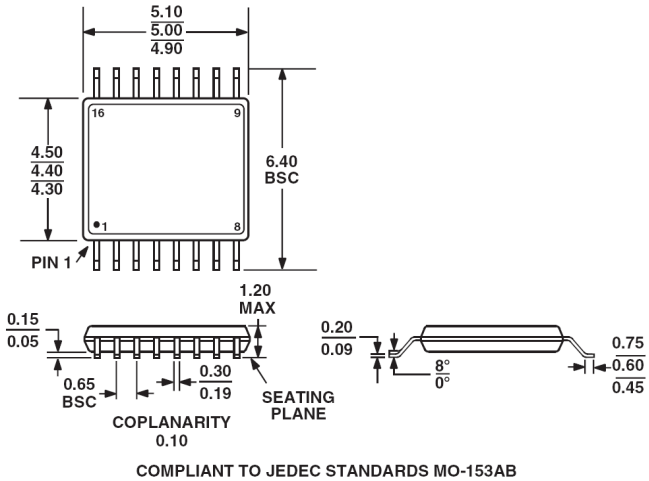
The AD5328 are octal 12-bit buffered voltage output DACs in a 16-lead TSSOP.
They operate from a single 2.5 V to 5.5 V supply, consuming 0.7 mA typ at 3 V.
Their on-chip output amplifiers allow the outputs to swing rail-to-rail with a slew rate of 0.7 V/µs.
The AD5328 use a versatile 3-wire serial interface that operates at clock rates up to 30 MHz and is compatible with standard SPI, QSPI, MICROWIRE, and DSP interface standards.
The references for the eight DACs are derived from two reference

o Functional Block Diagram



o Pin configuration and outline dimensions





o Pin function descriptions

Pin No.	Mnemonic	Function
1	LDAC	This active low-control input transfers the contents of the input registers to their respective DAC registers. Pulsing this pin low allows any or all DAC registers to be updated if the input registers have new data. This allows simultaneous update of all DAC outputs. Alternatively, this pin can be tied permanently low.
2	SYNC	Active Low-Control Input. This is the frame synchronization signal for the input data. When SYNC goes low, it powers on the SCLK and DIN buffers and enables the input shift register. Data is transferred in on the falling edges of the following 16 clocks. If SYNC is taken high before the 16th falling edge, the rising edge of SYNC acts as an interrupt and the write sequence is ignored by the device.
3	VDD	Power Supply Input. These parts can be operated from 2.5 V to 5.5 V, and the supply should be decoupled with a 10 µF capacitor in parallel with a 0.1 µF capacitor to GND.
4	VOUTA	Buffered Analog Output Voltage from DAC A. The output amplifier has rail-to-rail operation.
5	VOUTB	Buffered Analog Output Voltage from DAC B. The output amplifier has rail-to-rail operation.
6	VOUTC	Buffered Analog Output Voltage from DAC C. The output amplifier has rail-to-rail operation.
7	VOUTD	Buffered Analog Output Voltage from DAC D. The output amplifier has rail-to-rail operation.
8	VREFABCD	Reference Input Pin for DACs A, B, C, and D. It may be configured as a buffered, unbuffered, or VDD input to the four DACs, depending on the state of the BUF and VDD control bits. It has an input range from 0.25 V to VDD in unbuffered mode and from 1 V to VDD in buffered mode.
9	VREFEFGH	Reference Input Pin for DACs E, F, G, and H. It may be configured as a buffered, unbuffered, or VDD input to the four DACs, depending on the state of the BUF and VDD control bits. It has an input range from 0.25 V to VDD in unbuffered mode and from 1 V to VDD in buffered mode.
10	VOUTE	Buffered Analog Output Voltage from DAC E. The output amplifier has rail-to-rail operation.
11	VOUTF	Buffered Analog Output Voltage from DAC F. The output amplifier has rail-to-rail operation.
12	VOUTG	Buffered Analog Output Voltage from DAC G. The output amplifier has rail-to-rail operation.
13	VOUTH	Buffered Analog Output Voltage from DAC H. The output amplifier has rail-to-rail operation.
14	GND	Ground Reference Point for All Circuitry on the Part.
15	DIN	Serial Data Input. This device has a 16-bit shift register. Data is clocked into the register on the falling edge of the serial clock input. The DIN input buffer is powered down after each write cycle.
16	SCLK	Serial Clock Input. Data is clocked into the input shift register on the falling edge of the serial clock input. Data can be transferred at rates up to 30 MHz. The SCLK input buffer is powered down after each write cycle.

- **Power on reset**

The AD5308/AD5318/AD5328 are provided with a power-on reset function so that they power up in a defined state. The power-on state is

- Normal operation
- Reference inputs unbuffered
- 0 V to VREF output range
- Output voltage set to 0 V
- LDAC bits set to LDAC high

Both input and DAC registers are filled with zeros and remain so until a valid write sequence is made to the device. This is particularly useful in applications where it is important to know the state of the DAC outputs while the device is powering up.

- **Input Shift Register**

The input shift register is 16 bits wide. Data is loaded into the device as a 16-bit word under the control of a serial clock input, SCLK. The timing diagram for this operation is shown in Figure 11.
The SYNC input is a level-triggered input that acts as a frame synchronization signal and chip enable. Data can be transferred into the device only while SYNC is low.
To start the serial data transfer, SYNC should be taken low, observing the minimum SYNC to SCLK falling edge setup time, t4.
After SYNC goes low, serial data will be shifted into the device's input shift register on the falling edges of SCLK for 16 clock pulses.
To end the transfer, SYNC must be taken high after the falling edge of the 16th SCLK pulse, observing the minimum SCLK falling edge to SYNC rising edge time, t7.
After the end of serial data transfer, data will automatically be transferred from the input shift register to the input register of the selected DAC.
If SYNC is taken high before the 16th falling edge of SCLK, the data transfer will be aborted and the DAC input registers will not be updated.
Data is loaded MSB first (Bit 15). The first bit determines whether it is a DAC write or a control function.

- **DAC Write**

Here, the 16-bit word consists of one control bit and three address bits followed by 12 bits of DAC data.
In the case of a DAC write, the MSB will be a 0. The next three address bits determine whether the data is for DAC A, DAC B, DAC C, DAC D, DAC E, DAC F, DAC G, or DAC H.
The AD5328 uses all 12 bits of DAC data. The data format is straight binary, with all 0s corresponding to 0 V output and all 1s corresponding to full-scale output.

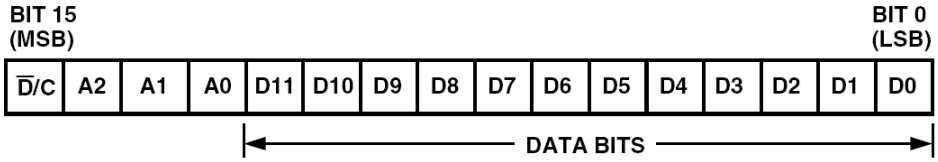


Figure 10 : AD5328 input shift register contents

- **Timing characteristics**

Parameter	A, B Version Limit at TMIN, TMAX	Unit	Conditions/Comments
t1	33 ns	min	SCLK Cycle Time
t2	13 ns	min	SCLK High Time
t3	13 ns	min	SCLK Low Time
t4	13 ns	min	SYNC to SCLK Falling Edge Setup Time
t5	5 ns	min	Data Setup Time
t6	4.5 ns	min	Data Hold Time

Parameter	A, B Version Limit at TMIN, TMAX	Unit	Conditions/Comments
t7	0 ns	min	SCLK Falling Edge to SYNC Rising Edge
t8	50 ns	min	Minimum SYNC High Time
t9	20 ns	min	LDAC Pulse width
t10	20 ns	min	SCLK Falling Edge to LDAC Rising Edge
t11	0 ns	min	SCLK Falling Edge to LDAC Falling Edge

NOTES

- ✓ Guaranteed by design and characterization; not production tested.
- ✓ All input signals are specified with tr = tf = 5 ns (10% to 90% of VDD) and timed from a voltage level of (VIL + VIH)/2.

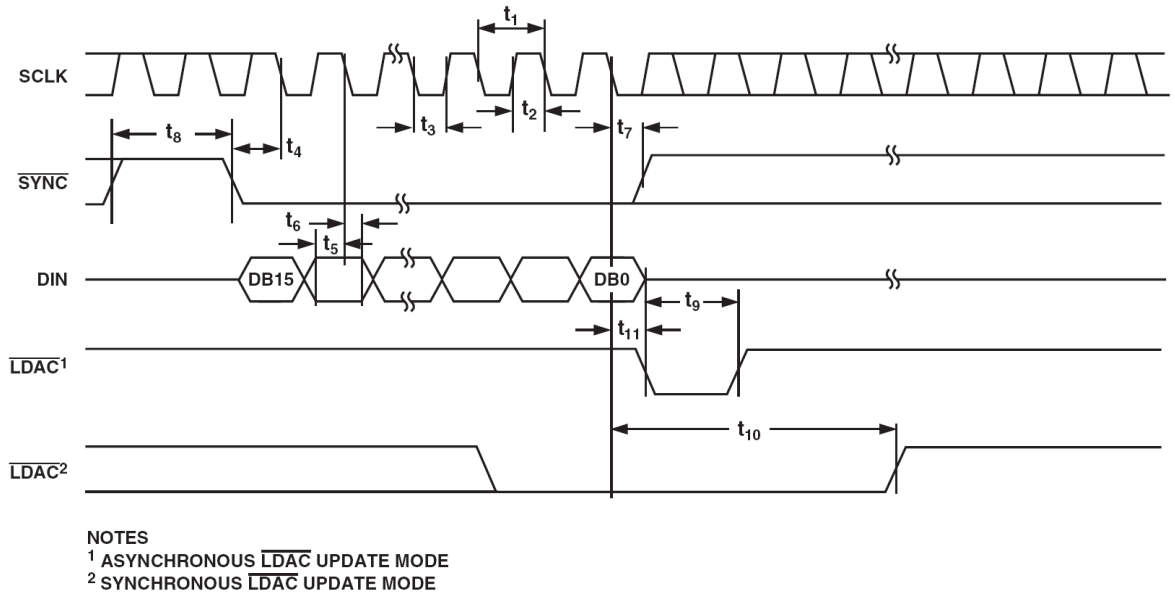
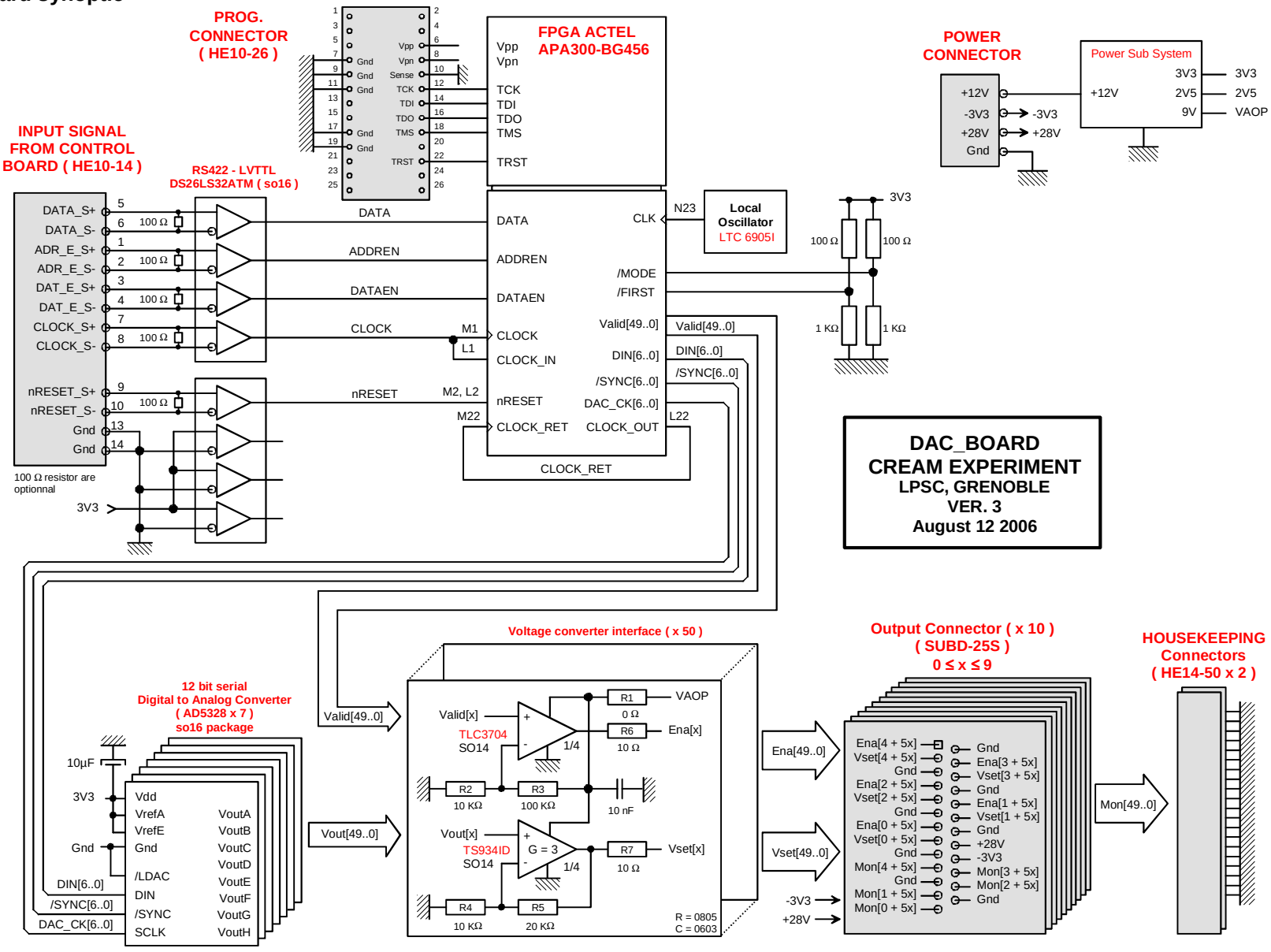


Figure 11 : Serial interface timing diagram

Annex 1 : Board synoptic



Annex 2 : HT module numeration for command and for housekeeping

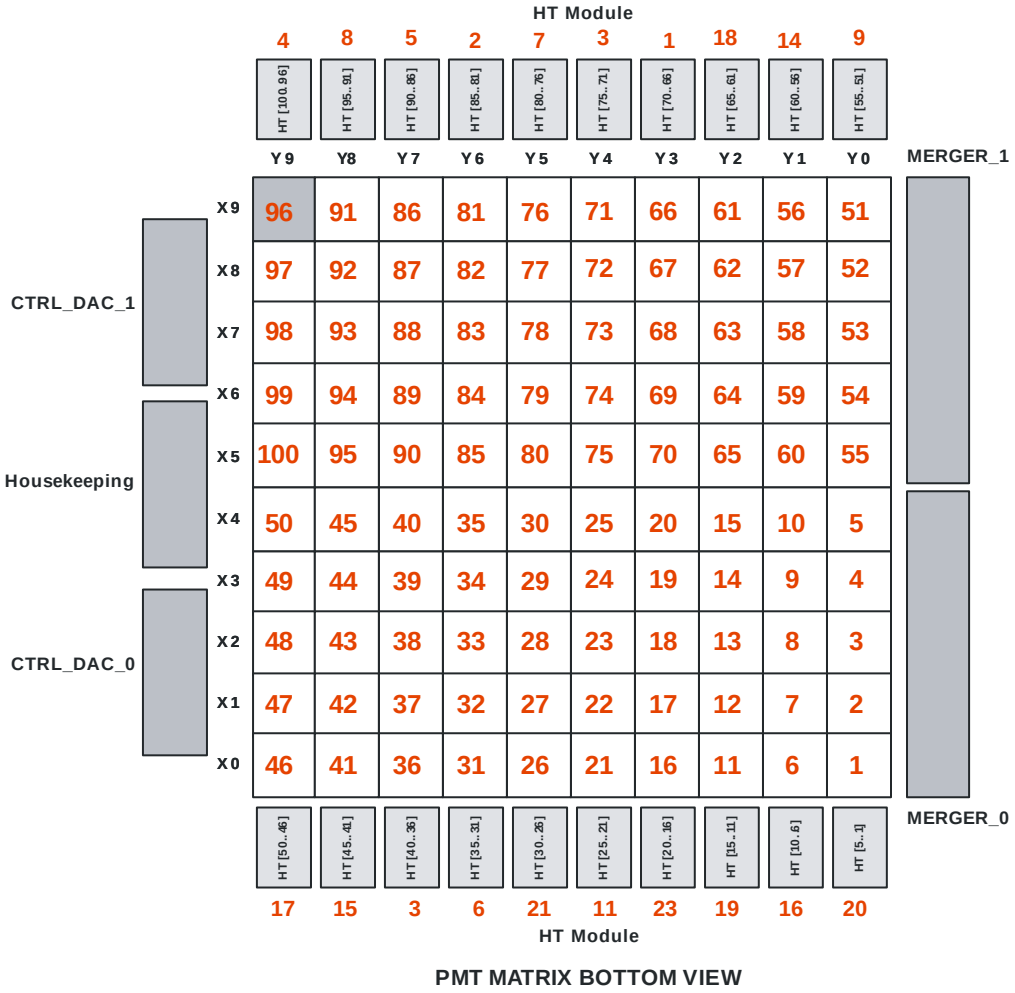


Figure 12 : HT module numeration for command

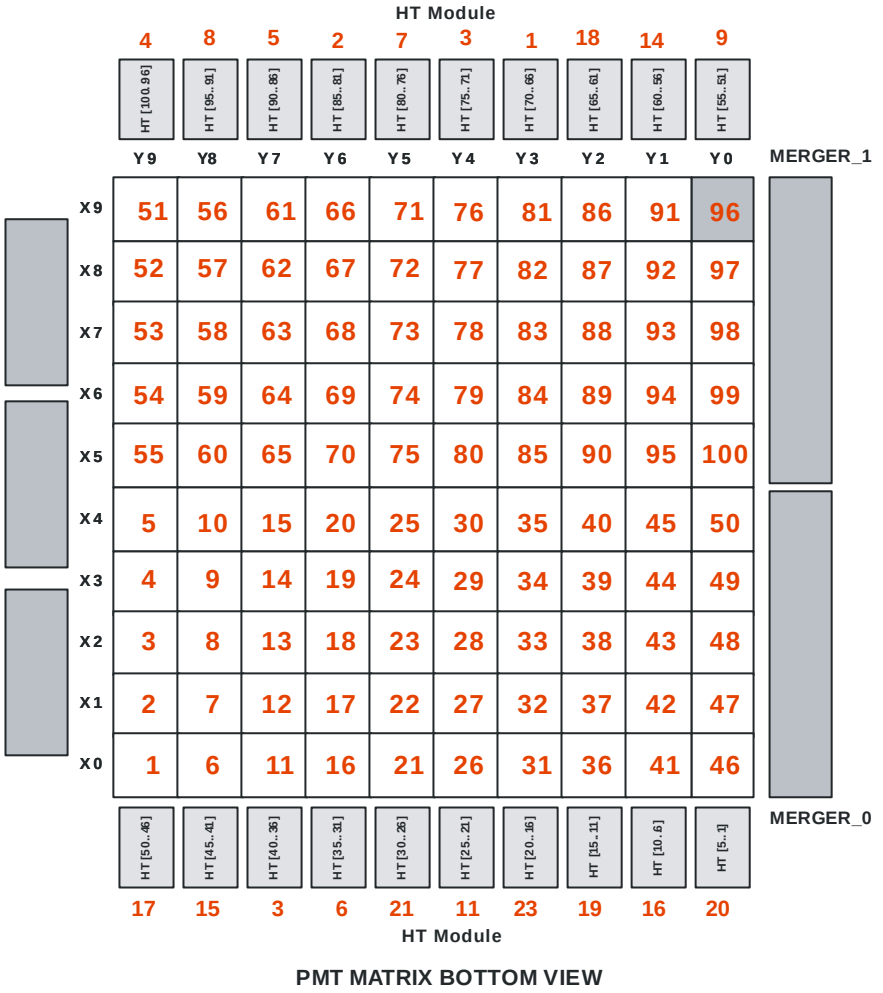


Figure 13 : HT module numeration for housekeeping

Annex 3 : FPGA Pin report (list per name)

Pin Report - Date: Thu Aug 09 14:35:23 2007
Pincheckcsum: NOT-AVAILABLE
Product: Designer
Release: v7.0
Version: 7.0.0.11
Design Name: cdac
Family: pa Package: 456 BGA

Port	Pin
ADDREN	N1
CLOCK	M22
CLOCK_IN	M1
CLOCK_OUT	L22
dac_ck[0]	R26
dac_ck[1]	T26
dac_ck[2]	U26
dac_ck[3]	V26
dac_ck[4]	W26
dac_ck[5]	Y26
dac_ck[6]	AA26
DATA	R1
DATA_OUT[0]	B8
DATA_OUT[1]	B9
DATA_OUT[2]	B10
DATA_OUT[3]	B11
DATA_OUT[4]	B12
DATA_OUT[5]	B13
DATA_OUT[6]	B14
DATAEN	P1
NCS[0]	A8
NCS[1]	A9
NCS[2]	A10
NCS[3]	A11
NCS[4]	A12
NCS[5]	A13
NCS[6]	A14
nFIRST	F1
VALID[45]	AC8
VALID[46]	AE8
VALID[47]	AD8
VALID[48]	AD7
VALID[49]	AC7

Port	Pin
nMODE	G1
nRESET	M2
test[1]	E1
test[2]	K1
test[3]	J1
test[4]	H1
test[5]	Y1
test[6]	W1
test[7]	V1
test[8]	U1
test[9]	T1
VALID[0]	AC19
VALID[1]	AC18
VALID[2]	AB19
VALID[3]	AC20
VALID[4]	AD19
VALID[5]	AD18
VALID[6]	AC17
VALID[7]	AD17
VALID[8]	AE19
VALID[9]	AE18
VALID[10]	AC16
VALID[11]	AD16
VALID[12]	AE17
VALID[13]	AF17
VALID[14]	AC15
VALID[15]	AD15
VALID[16]	AE16

Port	Pin
VALID[17]	AF16
VALID[18]	AD14
VALID[19]	AC13
VALID[20]	AE15
VALID[21]	AF15
VALID[22]	AD13
VALID[23]	AC12
VALID[24]	AE14
VALID[25]	AF14
VALID[26]	AB11
VALID[27]	AD12
VALID[28]	AE13
VALID[29]	AF13
VALID[30]	AE12
VALID[31]	AF12
VALID[32]	AC11
VALID[33]	AD11
VALID[34]	AF11
VALID[35]	AE11
VALID[36]	AC10
VALID[37]	AD10
VALID[38]	AF10
VALID[39]	AE10
VALID[40]	AD9
VALID[41]	AC9
VALID[42]	AF9
VALID[43]	AE9
VALID[44]	AB8

Annex 4 : FPGA Pin report (list per pin)

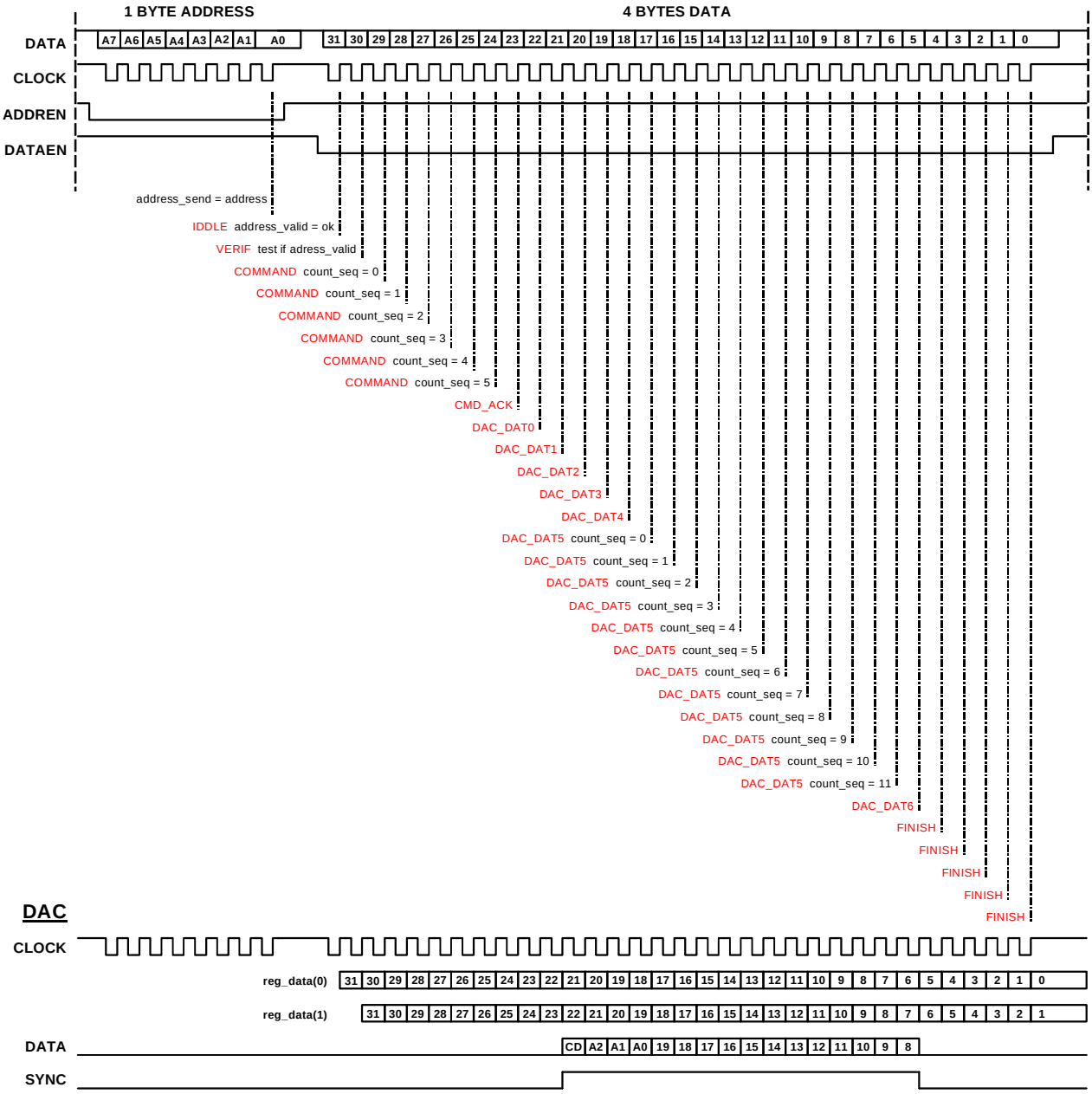
Pin Report - Date: Thu Aug 09 14:35:55 2007
Pincheckcsum: NOT-AVAILABLE
Product: Designer
Release: v7.0
Version: 7.0.0.11
Design Name: cdac
Family: pa Package: 456 BGA

Number	Port
A8	NCS[0]
A9	NCS[1]
A10	NCS[2]
A11	NCS[3]
A12	NCS[4]
A13	NCS[5]
A14	NCS[6]
AA26	dac_ck[6]
AB8	VALID[44]
AB11	VALID[26]
AB19	VALID[2]
AC7	VALID[49]
AC8	VALID[45]
AC9	VALID[41]
AC10	VALID[36]
AC11	VALID[32]
AC12	VALID[23]
AC13	VALID[19]
AC15	VALID[14]
AC16	VALID[10]
AC17	VALID[6]
AC18	VALID[1]
AC19	VALID[0]
AC20	VALID[3]
AD7	VALID[48]
AD8	VALID[47]
AD9	VALID[40]
AD10	VALID[37]
AD11	VALID[33]
AD12	VALID[27]

Number	Port
AD13	VALID[22]
AD14	VALID[18]
AD15	VALID[15]
AD16	VALID[11]
AD17	VALID[7]
AD18	VALID[5]
AD19	VALID[4]
AE8	VALID[46]
AE9	VALID[43]
AE10	VALID[39]
AE11	VALID[35]
AE12	VALID[30]
AE13	VALID[28]
AE14	VALID[24]
AE15	VALID[20]
AE16	VALID[16]
AE17	VALID[12]
AE18	VALID[9]
AE19	VALID[8]
AF9	VALID[42]
AF10	VALID[38]
AF11	VALID[34]
AF12	VALID[31]
AF13	VALID[29]
AF14	VALID[25]
AF15	VALID[21]
AF16	VALID[17]
AF17	VALID[13]
B8	DATA_OUT[0]
B9	DATA_OUT[1]

Number	Port
B10	DATA_OUT[2]
B11	DATA_OUT[3]
B12	DATA_OUT[4]
B13	DATA_OUT[5]
B14	DATA_OUT[6]
E1	test[1]
F1	nFIRST
G1	nMODE
H1	test[4]
J1	test[3]
K1	test[2]
L22	CLOCK_OUT
M1	CLOCK_IN
M2	nRESET
M22	CLOCK
N1	ADDREN
P1	DATAEN
R1	DATA
R26	dac_ck[0]
T1	test[9]
T26	dac_ck[1]
U1	test[8]
U26	dac_ck[2]
V1	test[7]
V26	dac_ck[3]
W1	test[6]
W26	dac_ck[4]
Y1	test[5]
Y26	dac_ck[5]

Annex 5 : FPGA signal generation



Annex 6 : FPGA Power consumption

Calculating Typical Power Dissipation

ProASICPLUS device power is calculated with both a static and an active component. The active component is a function of both the number of tiles utilized and the system speed.
Power dissipation can be calculated using the following formula:

Total Power Consumption—Ptotal

Ptotal = Pdc + Pac

where:

Pdc = 11 mW for the APA300
Pdc includes the static components of PVDDP + PVDD + PAVDD
Pac = Pclock + Pstorage + Plogic + Poutputs + Pinputs + Ppll + Pmemory

Global Clock Contribution—Pclock

Pclock, the clock component of power dissipation, is given by the piece-wise model:

for R < 15000 the model is: (P1 + (P2*R) – (P7*R2)) * Fs (lightly-loaded clock trees)
for R > 15000 the model is: (P10 + P11*R) * Fs (heavily-loaded clock trees)

where:

P1 = 100 µW/MHz is the basic power consumption of the clock tree per MHz of the clock
P2 = 1.3 µW/MHz is the incremental power consumption of the clock tree per storage tile – also per MHz of the clock
P7 = 0.00003 µW/MHz is a correction factor for partially-loaded clock trees
P10 = 6850 µW/MHz is the basic power consumption of the clock tree per MHz of the clock
P11 = 0.4 µW/MHz is the incremental power consumption of the clock tree per storage tile – also per MHz of the clock
R = the number of storage tiles clocked by this clock
Fs = the clock frequency

Storage-Tile Contribution—Pstorage

Pstorage, the storage-tile (Register) component of AC power dissipation, is given by
Pstorage = P5 * ms * Fs
where:

P5 = 1.1 µW/MHz is the average power consumption of a storage tile per MHz of its output toggling rate. The maximum output toggling rate is Fs/2.
Ms = the number of storage tiles (Register) switching during each Fs cycle
Fs = the clock frequency

Logic-Tile Contribution—Plogic

Plogic, the logic-tile component of AC power dissipation, is given by
Plogic = P3 * mc * Fs
where:

P3 = 1.4 µW/MHz is the average power consumption of a logic tile per MHz of its output toggling rate. The maximum output toggling rate is Fs/2.
Mc = the number of logic tiles switching during each Fs cycle
Fs = the clock frequency

I/O Output Buffer Contribution—Poutputs

Poutputs, the I/O component of AC power dissipation, is given by
Poutputs = (P4 + (Cload * VDDP2)) * p * Fp
where:

P4 = 326 µW/MHz is the intrinsic power consumption of an output pad normalized per MHz of the output frequency. This is the total I/O current VDDP.
Cload = the output load
p = the number of outputs
Fp = the average output frequency

I/O Input Buffer's Buffer Contribution—Pinputs

The input's component of AC power dissipation is given by

Pinputs = P8 * q * Fq

where:

P8 = 29 µW/MHz is the intrinsic power consumption of an input pad normalized per MHz of the input frequency.
Q = the number of inputs
Fq = the average input frequency

PLL Contribution—Ppll

Ppll = P9 * Npll

where:

P9 = 7.5 mW. This value has been estimated at maximum PLL clock frequency.
NPll = number of PLLs used

RAM Contribution—Pmemory

Finally, Pmemory, the memory component of AC power consumption, is given by

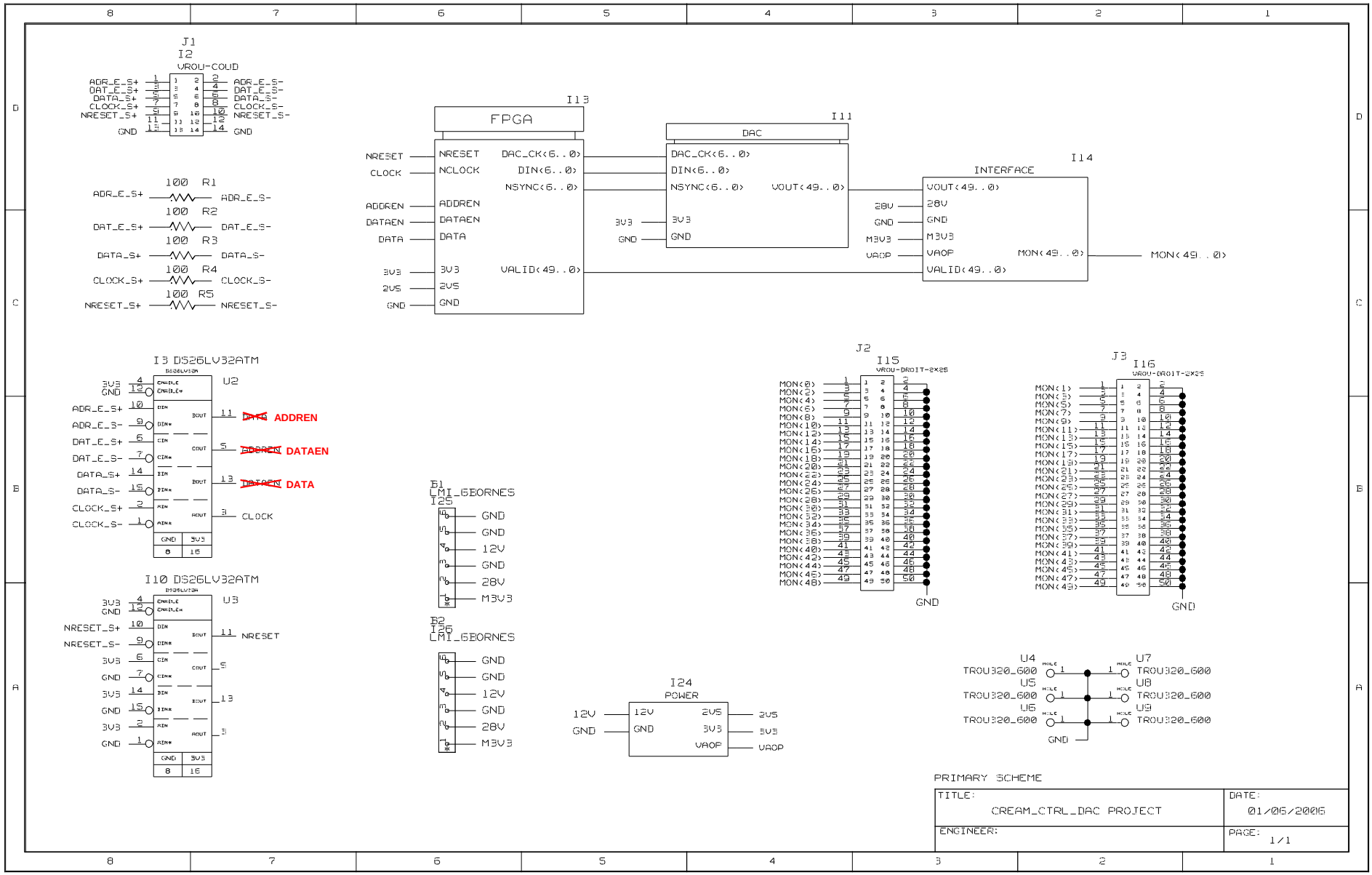
Pmemory = P6 * Nmemory * Fmemory * Ememory

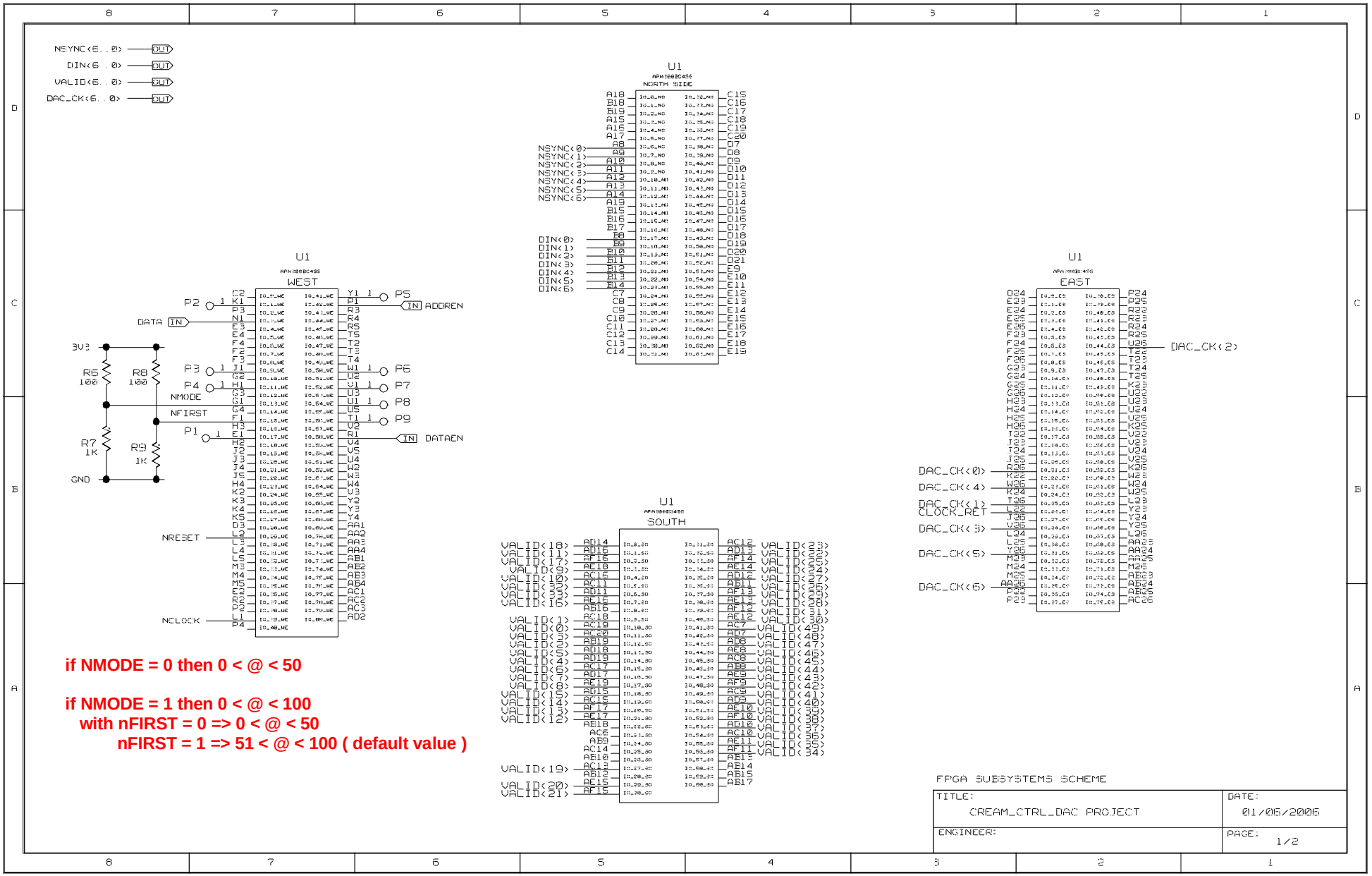
where:

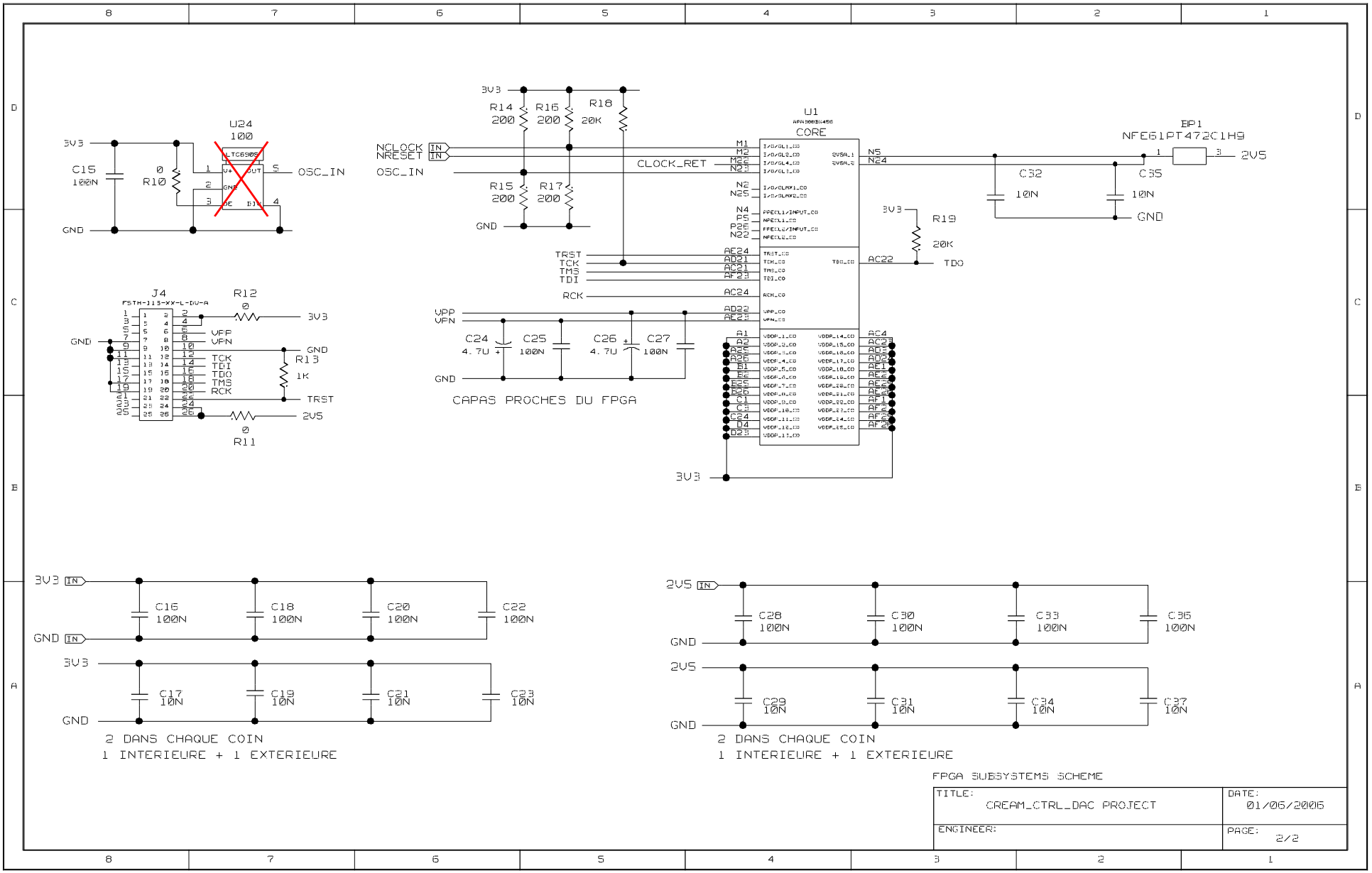
P6 = 175 µW/MHz is the average power consumption of a memory block per MHz of the clock
Nmemory = the number of RAM/FIFO blocks (1 block = 256 words * 9 bits)
Fmemory = the clock frequency of the memory
Ememory = the average number of active blocks divided by the total number of blocks (N) of the memory.

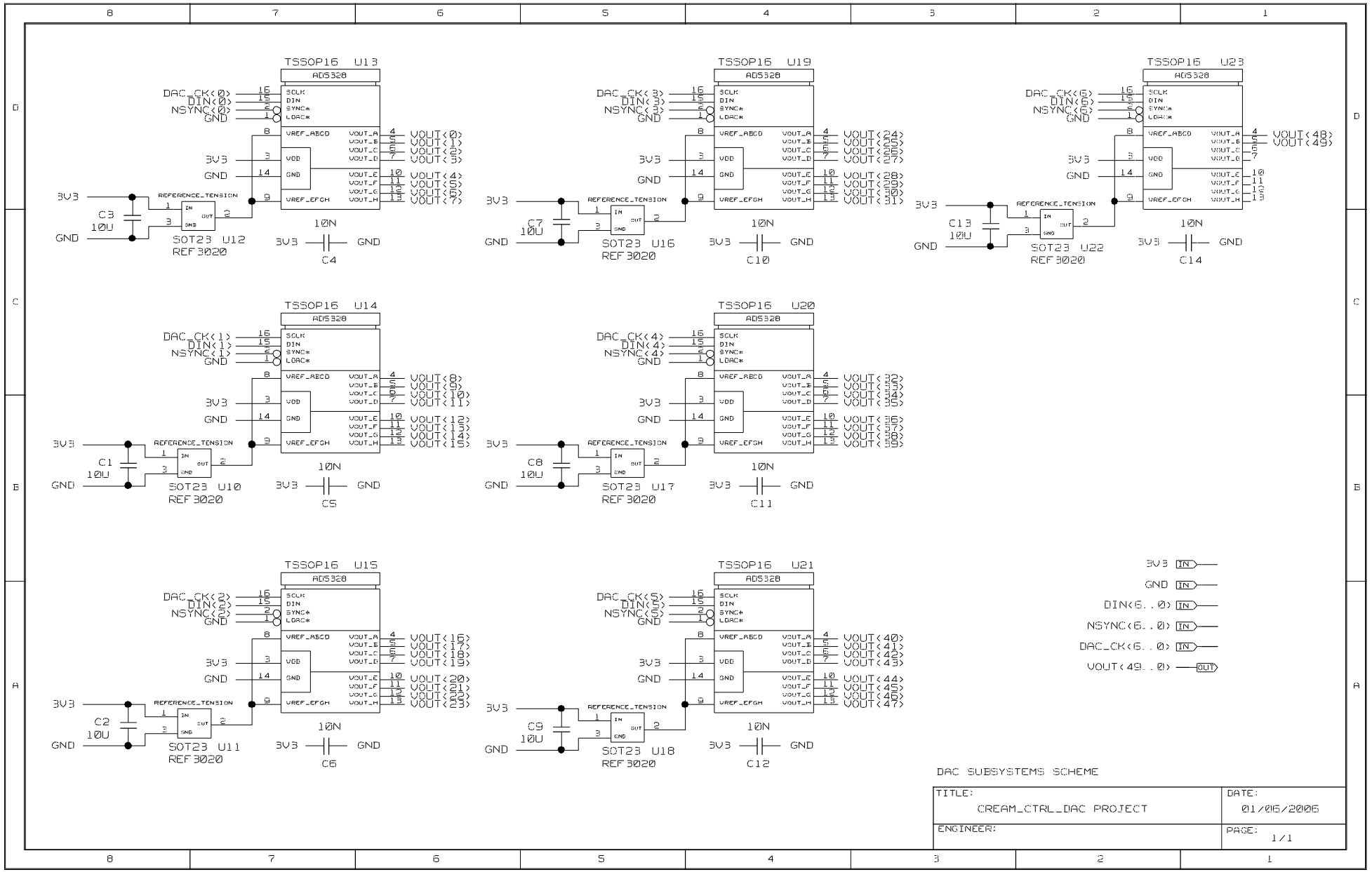
- Typical values for Ememory would be 1/4 for a 1k x 8,9,16, 32 memory and 1/16 for a 4kx8, 9, 16, and 32 memory configuration
- In addition, an application-dependent component to Ememory can be considered. For example, for a 1kx8 memory configuration using only 1 cycle out of 2, Ememory = 1/4*1/2 = 1/8

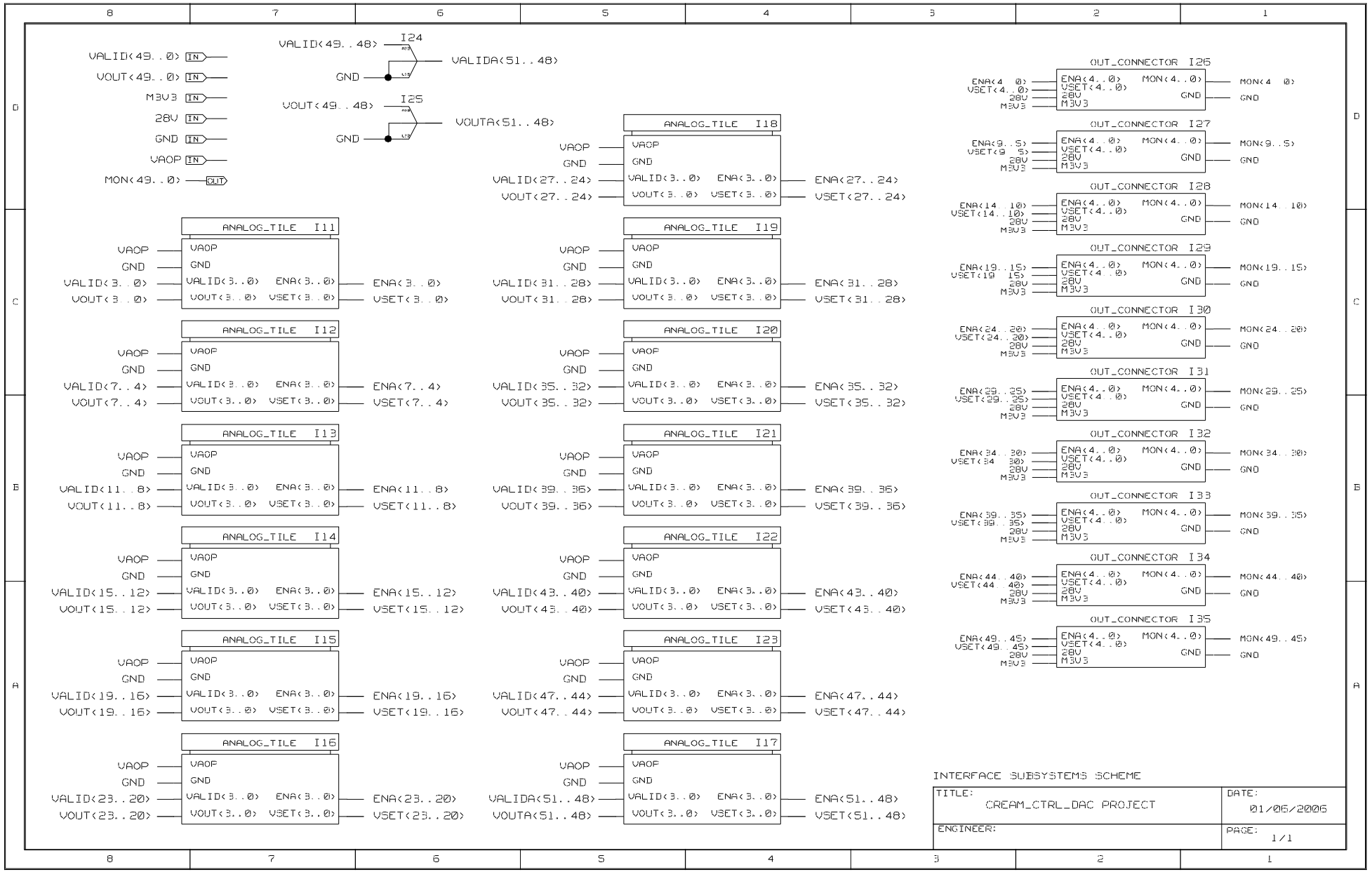
Annex 7 : Board Scheme

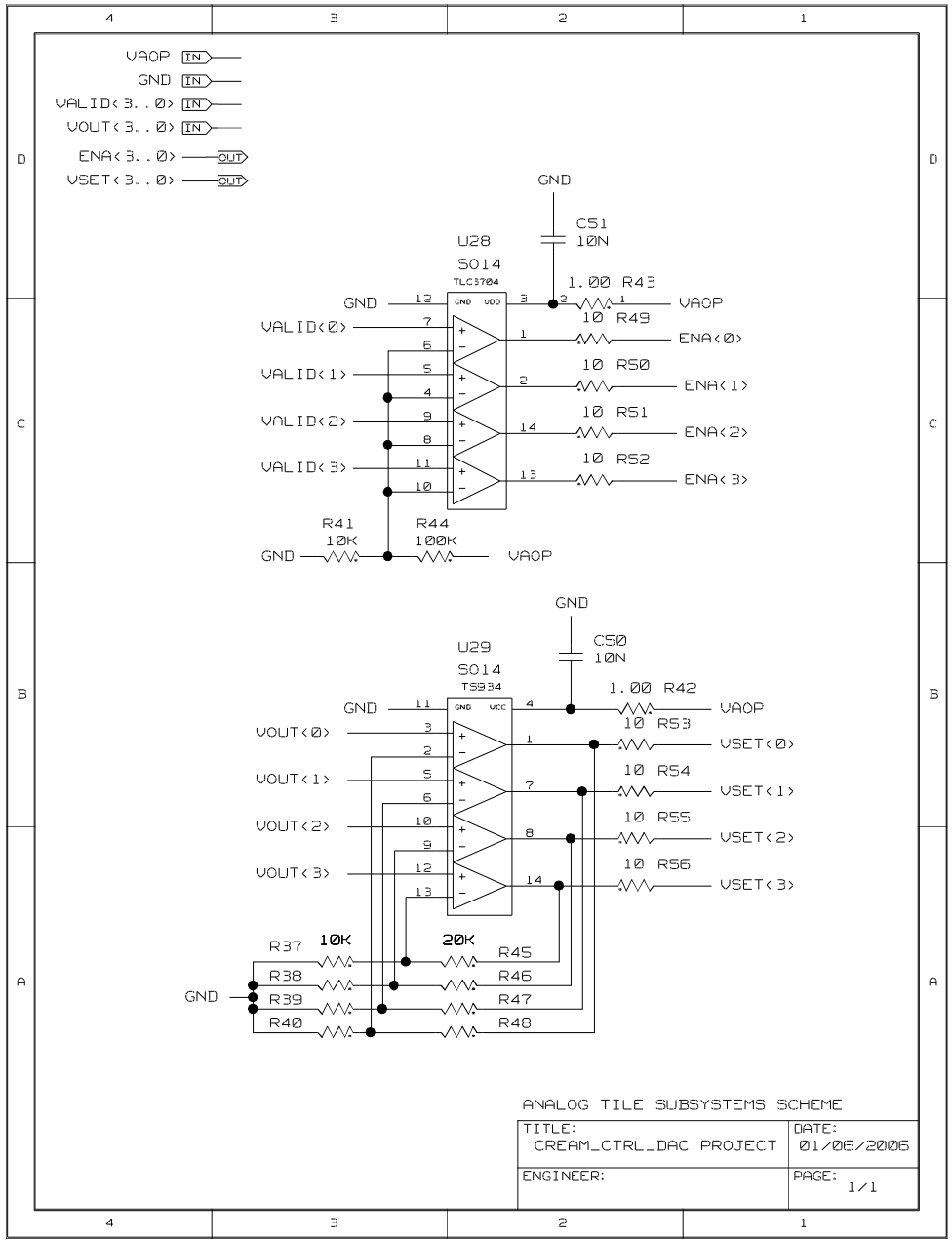


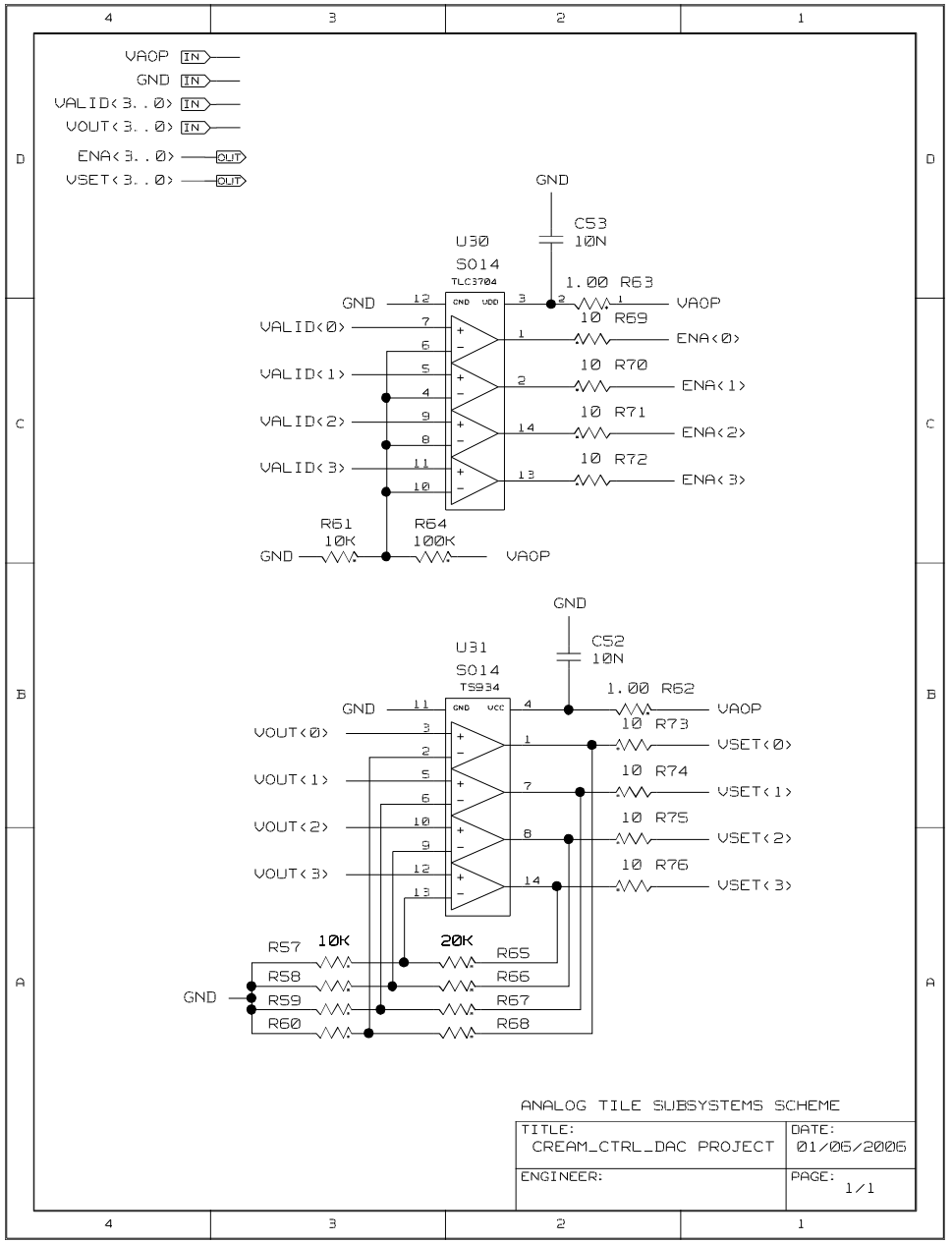

NSYNC<6..0>
DIN<6..0>
VALID<6..0>
DAC_CK<6..0>
OUT
U1
WEST
NORTH SIDE
SOUTH
EAST
P2
P3
P4
P1
P5
P6
P7
P8
P9
DATA
NMODE
NFIRST
NRESET
NCLOCK
ADDREN
DATAEN
3V3
GND
R5 100
R8 100
R7 1K
R9 1K
DAC_CK<2>
DAC_CK<0>
DAC_CK<4>
DAC_CK<1>
CLOCK_RET
DAC_CK<3>
DAC_CK<5>
DAC_CK<6>
if NMODE = 0 then 0 < @ < 50
if NMODE = 1 then 0 < @ < 100
with nFIRST = 0 => 0 < @ < 50
nFIRST = 1 => 51 < @ < 100 (default value)
FPGA SUBSYSTEMS SCHEME
TITLE:
CREAM_CTRL_DAC PROJECT
DATE:
01/05/2006
ENGINEER:
PAGE:
1/2

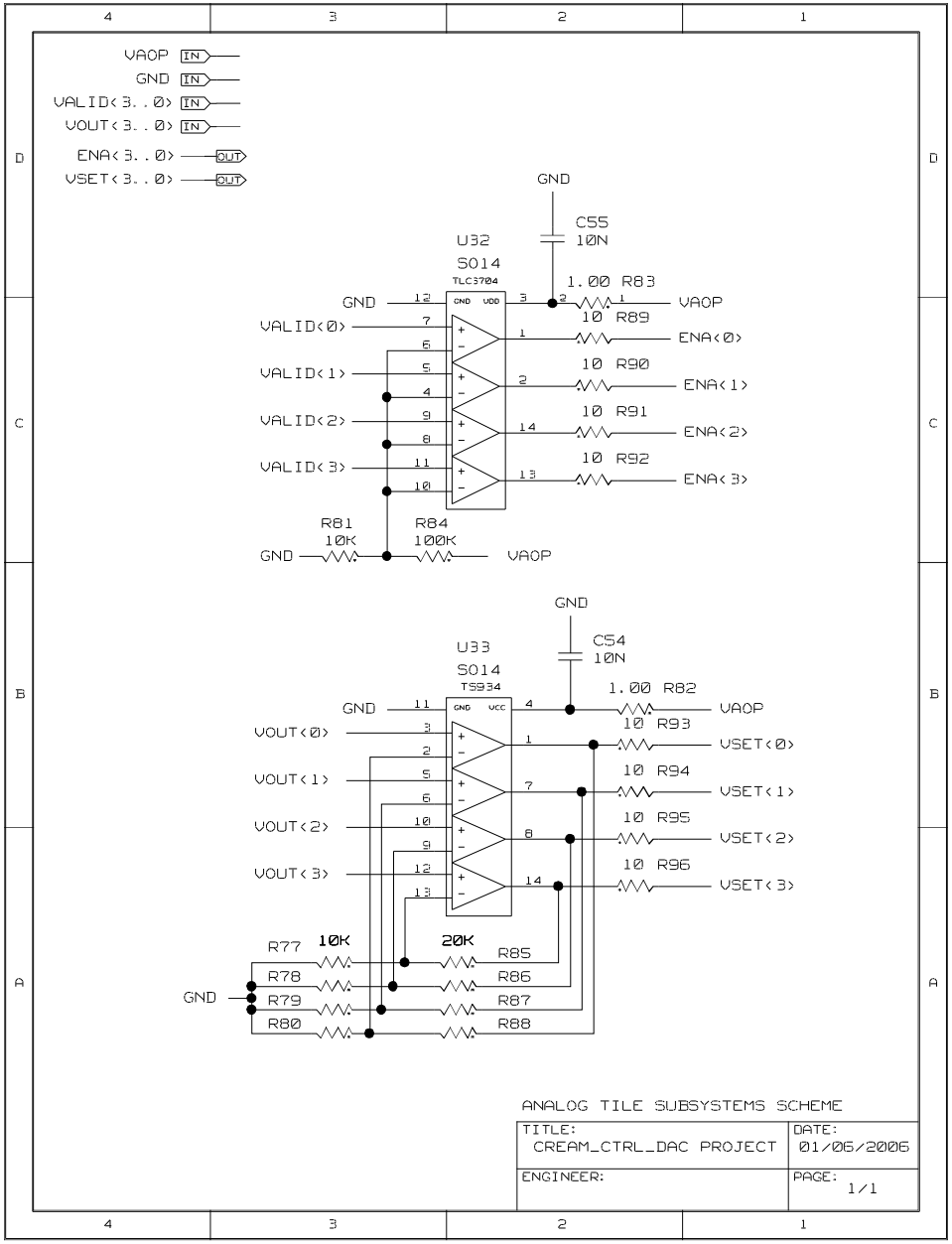

U24
100
3V3
C15
100N
R10
0
GND
OSC_IN
J4
FSTH-115-XX-L-DV-A
R12
0
3V3
VPP
VFN
GND
TCK
TDI
TDO
TMS
RCK
R13
1K
TRST
2V5
R11
0
U1
CORE
NCLOCK
NRESET
OSC_IN
CLOCK_RET
R14
200
R15
200
R18
20K
R17
200
TRST
TCK
TMS
TDI
RCK
VPP
VFN
C24
4.7U
C25
100N
C26
4.7U
C27
100N
CAPAS PROCHES DU FPGA
BP1
NFE61PT472C1H9
C32
10N
C35
10N
2V5
R19
20K
AC22
TDO
C16
100N
C18
100N
C20
100N
C22
100N
C17
10N
C19
10N
C21
10N
C23
10N
2 DANS CHAQUE COIN
1 INTERIEURE + 1 EXTERIEURE
C28
100N
C30
100N
C33
100N
C36
100N
C29
10N
C31
10N
C34
10N
C37
10N
2 DANS CHAQUE COIN
1 INTERIEURE + 1 EXTERIEURE
FPGA SUBSYSTEMS SCHEME
TITLE:
CREAM_CTRL_DAC PROJECT
DATE:
01/05/2005
ENGINEER:
PAGE:
2/2

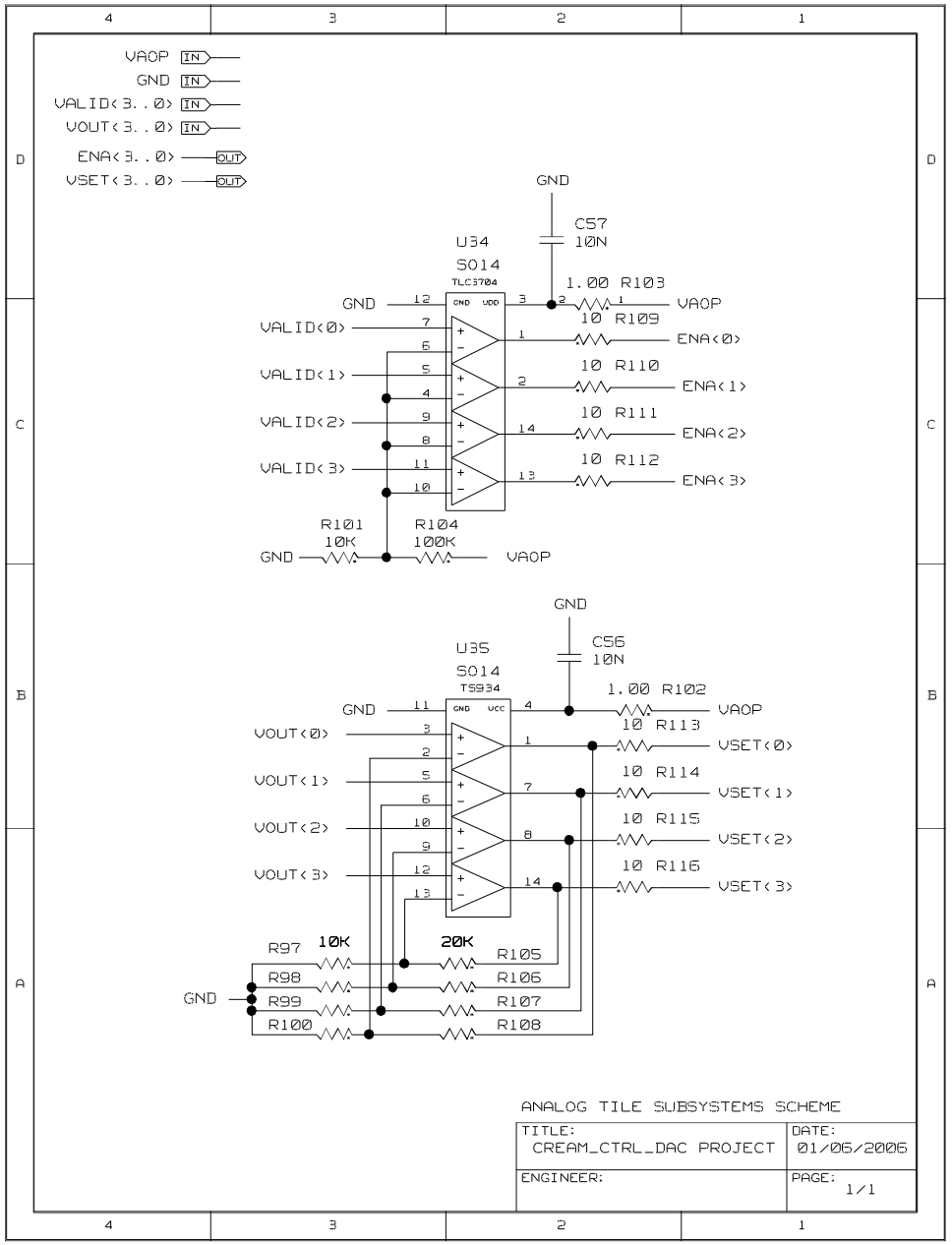

TSSOP16 U13
TSSOP16 U19
TSSOP16 U23
TSSOP16 U14
TSSOP16 U20
TSSOP16 U15
TSSOP16 U21
ADS328
SOT23 U12
SOT23 U16
SOT23 U22
SOT23 U10
SOT23 U17
SOT23 U11
SOT23 U18
REF3020
REFERENCE_TENSION
3V3
GND
10N
10U
C3
C4
C7
C10
C13
C14
C1
C5
C8
C11
C2
C6
C9
C12
DAC_CK<0>
DIN<0>
NSYNC<0>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
VOUT<4>
VOUT<5>
VOUT<6>
VOUT<7>
DAC_CK<3>
DIN<3>
NSYNC<3>
VOUT<24>
VOUT<25>
VOUT<26>
VOUT<27>
VOUT<28>
VOUT<29>
VOUT<30>
VOUT<31>
DAC_CK<6>
DIN<6>
NSYNC<6>
VOUT<48>
VOUT<49>
DAC_CK<1>
DIN<1>
NSYNC<1>
VOUT<8>
VOUT<9>
VOUT<10>
VOUT<11>
VOUT<12>
VOUT<13>
VOUT<14>
VOUT<15>
DAC_CK<4>
DIN<4>
NSYNC<4>
VOUT<32>
VOUT<33>
VOUT<34>
VOUT<35>
VOUT<36>
VOUT<37>
VOUT<38>
VOUT<39>
DAC_CK<2>
DIN<2>
NSYNC<2>
VOUT<16>
VOUT<17>
VOUT<18>
VOUT<19>
VOUT<20>
VOUT<21>
VOUT<22>
VOUT<23>
DAC_CK<5>
DIN<5>
NSYNC<5>
VOUT<40>
VOUT<41>
VOUT<42>
VOUT<43>
VOUT<44>
VOUT<45>
VOUT<46>
VOUT<47>
3V3 IN
GND IN
DIN<6..0> IN
NSYNC<6..0> IN
DAC_CK<6..0> IN
VOUT<49..0> OUT
DAC SUBSYSTEMS SCHEME
TITLE:
CREAM_CTRL_DAC PROJECT
DATE:
01/05/2006
ENGINEER:
PAGE:
1/1

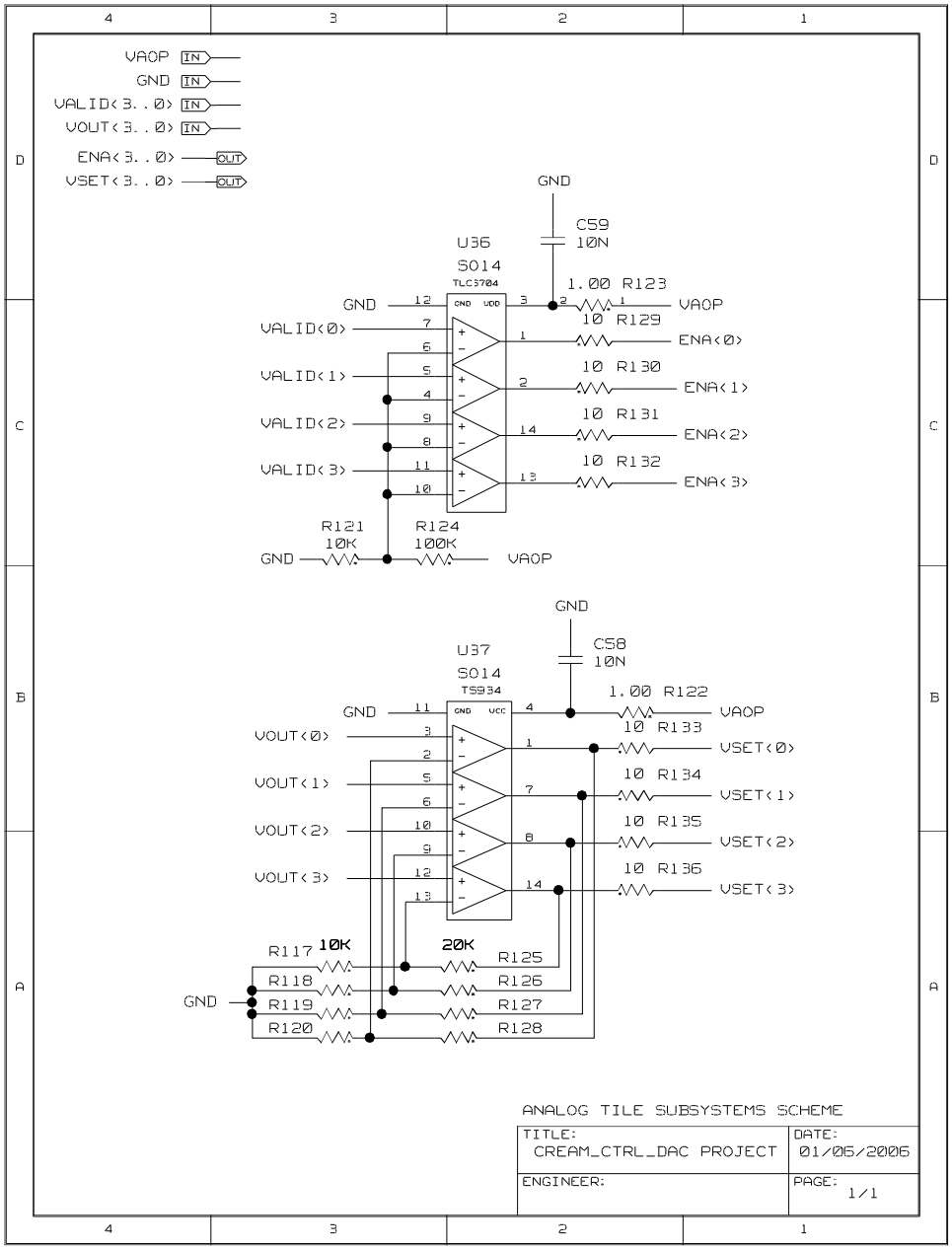

VALID<49..0> IN
VOUT<49..0> IN
M3V3 IN
28V IN
GND IN
VAOP IN
MON<49..0> OUT
I24
VALID<49..48>
GND
VALIDA<51..48>
I25
VOUT<49..48>
GND
VOUTA<51..48>
ANALOG_TILE I11
ANALOG_TILE I12
ANALOG_TILE I13
ANALOG_TILE I14
ANALOG_TILE I15
ANALOG_TILE I16
ANALOG_TILE I17
ANALOG_TILE I18
ANALOG_TILE I19
ANALOG_TILE I20
ANALOG_TILE I21
ANALOG_TILE I22
ANALOG_TILE I23
VAOP
GND
VALID<3..0>
VOUT<3..0>
ENA<3..0>
VSET<3..0>
OUT_CONNECTOR I26
OUT_CONNECTOR I27
OUT_CONNECTOR I28
OUT_CONNECTOR I29
OUT_CONNECTOR I30
OUT_CONNECTOR I31
OUT_CONNECTOR I32
OUT_CONNECTOR I33
OUT_CONNECTOR I34
OUT_CONNECTOR I35
ENA<4..0>
VSET<4..0>
28V
M3V3
MON<4..0>
INTERFACE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2005
ENGINEER:
PAGE: 1/1

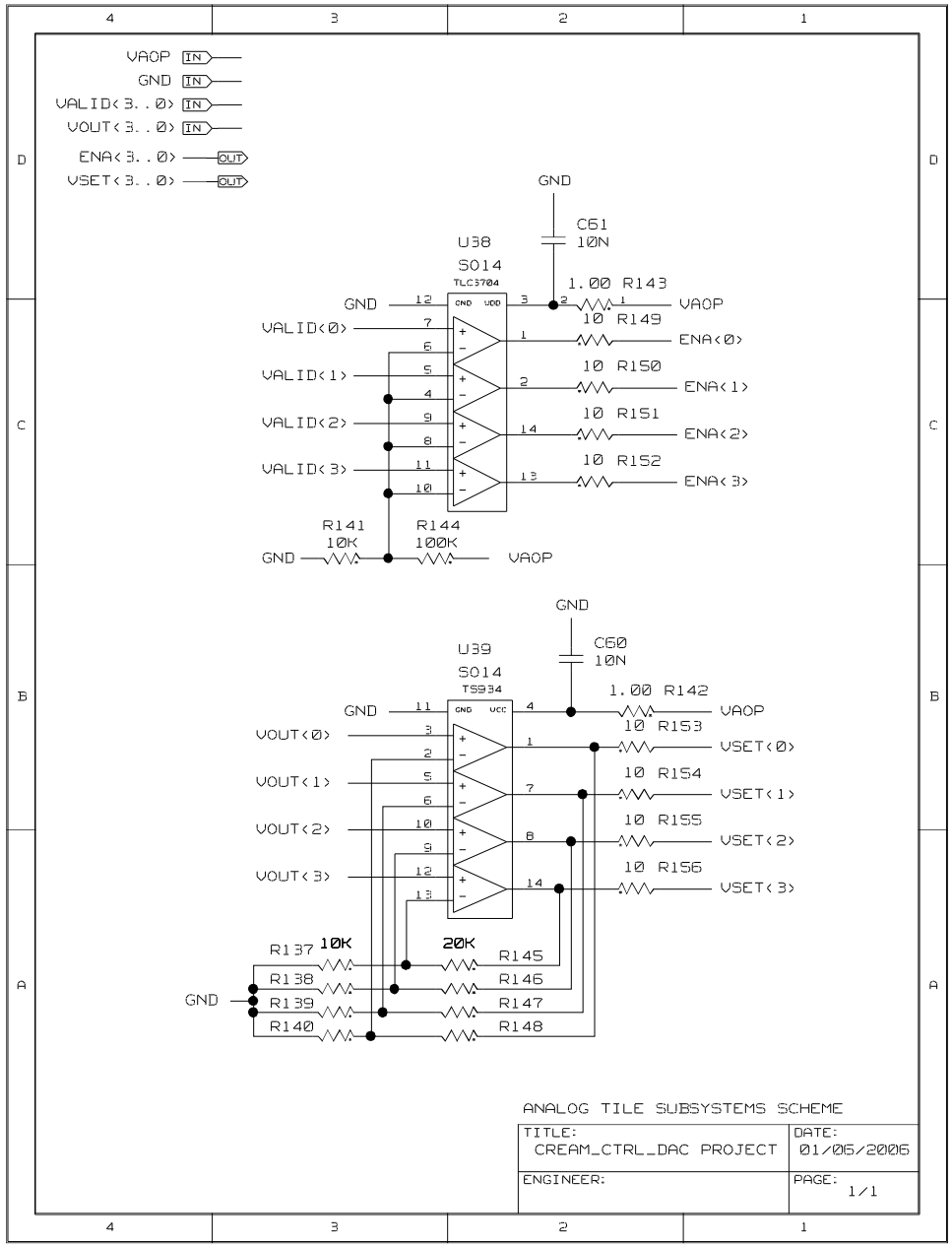

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C51
10N
U28
SO14
TLC3704
1.00 R43
10 R49
10 R50
10 R51
10 R52
VALID<0>
VALID<1>
VALID<2>
VALID<3>
ENA<0>
ENA<1>
ENA<2>
ENA<3>
R41
10K
R44
100K
U29
SO14
TS934
C50
10N
1.00 R42
10 R53
10 R54
10 R55
10 R56
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
VSET<0>
VSET<1>
VSET<2>
VSET<3>
R37 10K
20K
R45
R46
R47
R48
R38
R39
R40
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2006
ENGINEER:
PAGE: 1/1

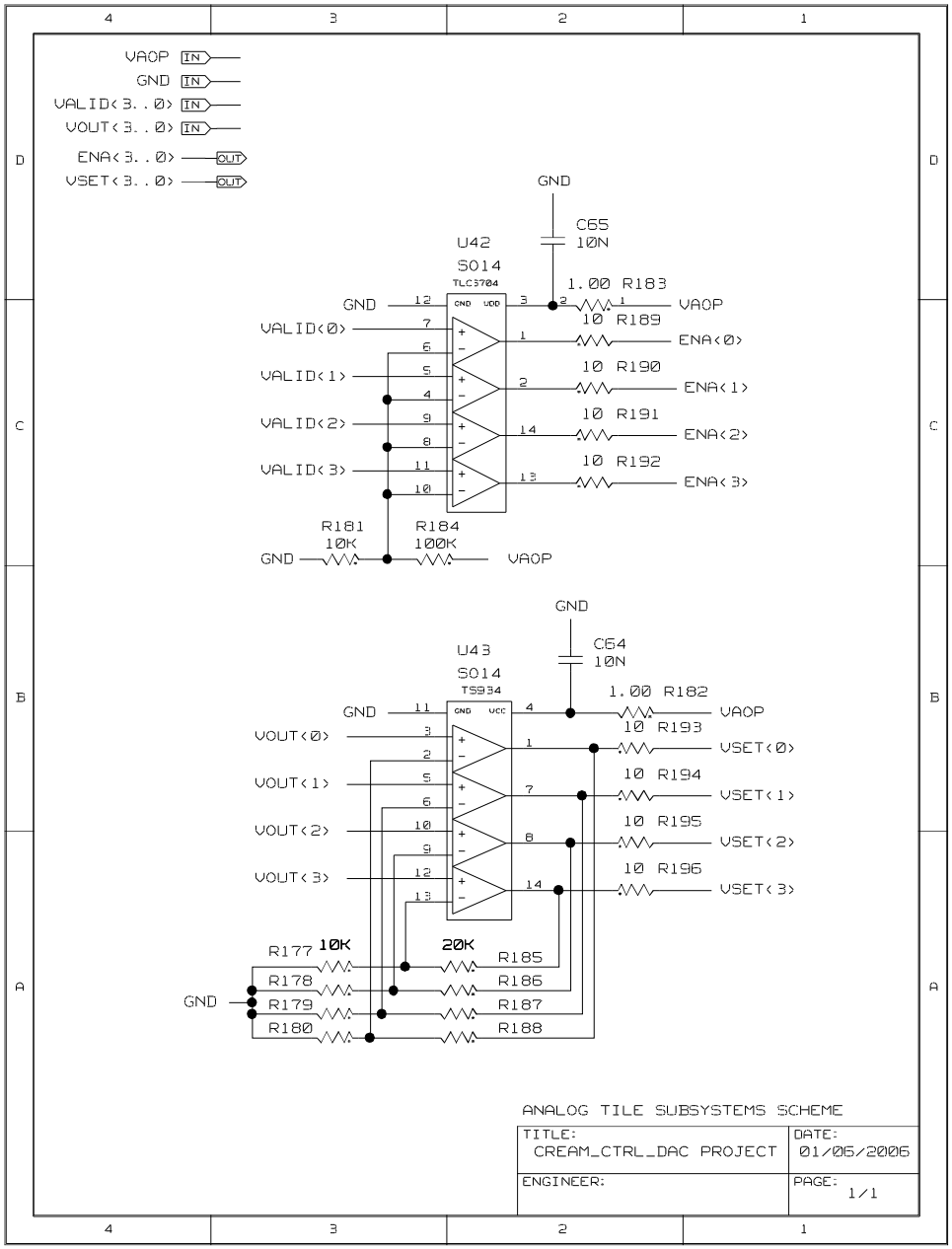

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C53
10N
U30
SO14
TLC3704
1.00 R63
10 R69
10 R70
10 R71
10 R72
VALID<0>
VALID<1>
VALID<2>
VALID<3>
ENA<0>
ENA<1>
ENA<2>
ENA<3>
R61
10K
R64
100K
U31
SO14
TS934
C52
10N
1.00 R62
10 R73
10 R74
10 R75
10 R76
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
VSET<0>
VSET<1>
VSET<2>
VSET<3>
R57 10K
20K
R65
R66
R67
R68
R58
R59
R60
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2006
ENGINEER:
PAGE: 1/1

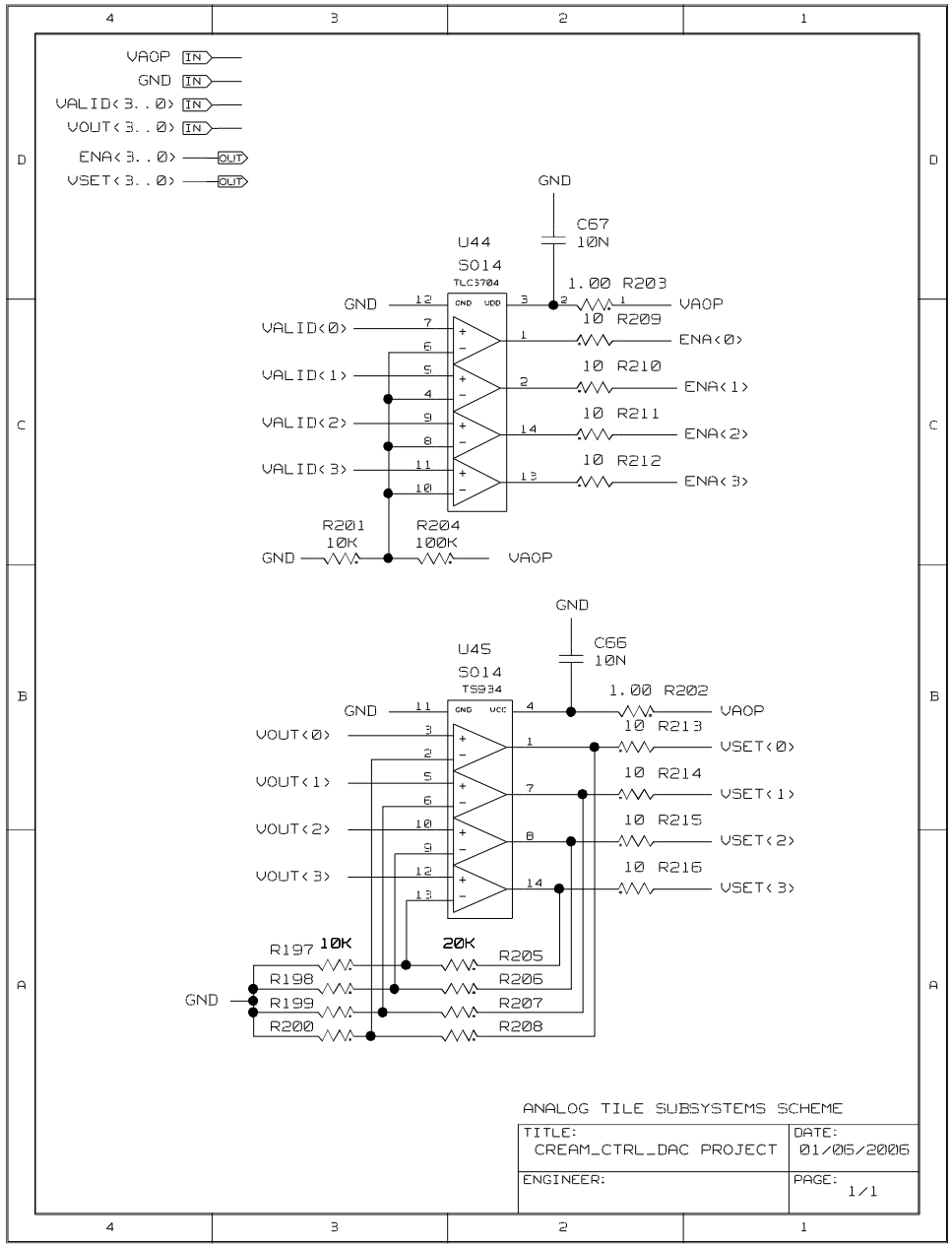

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C55
10N
U32
S014
TLC3704
1.00 R83
10 R89
10 R90
10 R91
10 R92
VAOP
ENA<0>
ENA<1>
ENA<2>
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R81
10K
R84
100K
U33
S014
TS934
C54
10N
1.00 R82
10 R93
10 R94
10 R95
10 R96
VSET<0>
VSET<1>
VSET<2>
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R77
10K
20K
R78
R79
R80
R85
R86
R87
R88
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2006
ENGINEER:
PAGE: 1/1

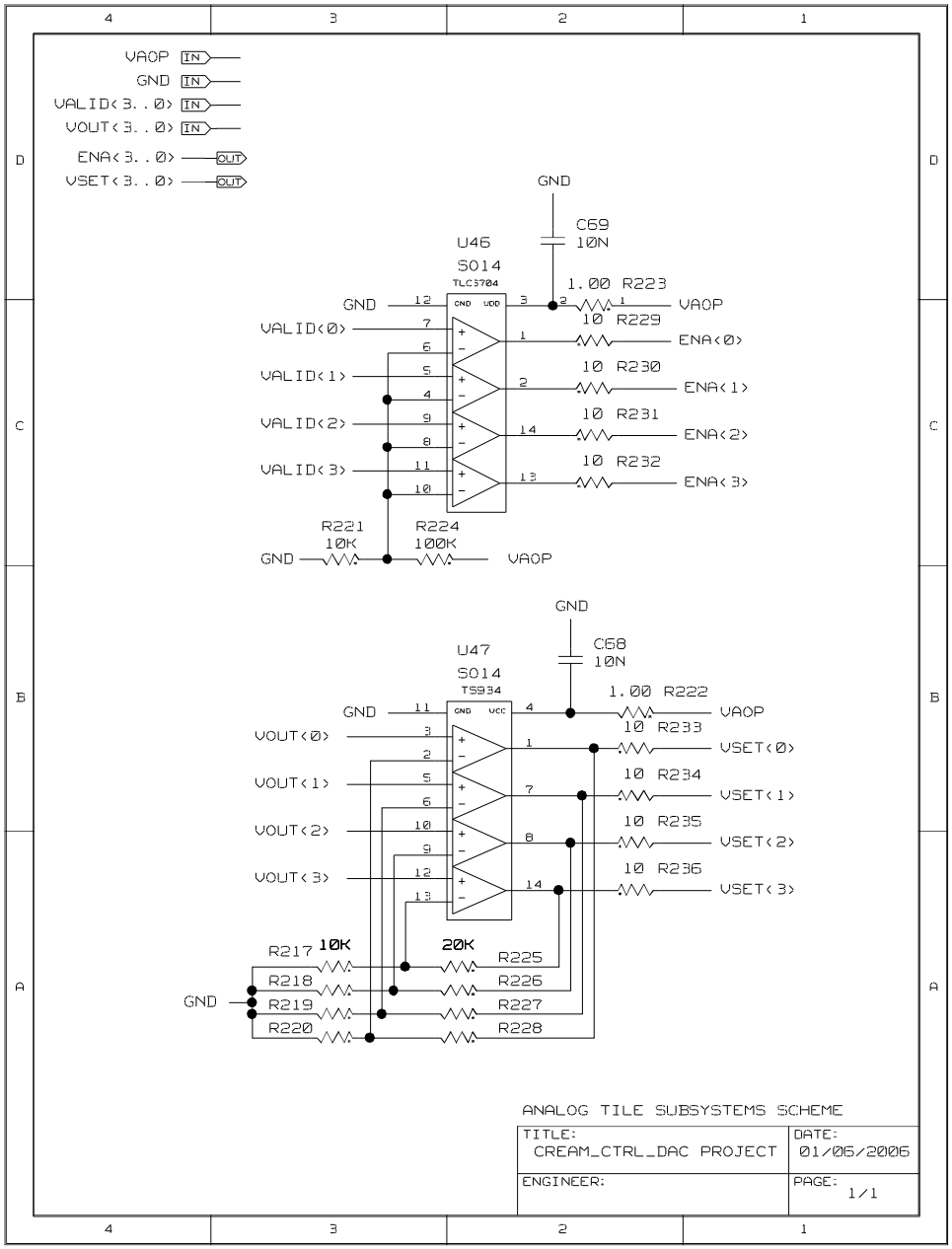

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C57
10N
U34
S014
TLC3704
1.00 R103
10 R109
10 R110
10 R111
10 R112
VAOP
ENA<0>
ENA<1>
ENA<2>
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R101
10K
R104
100K
U35
S014
TS934
C56
10N
1.00 R102
10 R113
10 R114
10 R115
10 R116
VSET<0>
VSET<1>
VSET<2>
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R97
10K
20K
R98
R99
R100
R105
R106
R107
R108
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2006
ENGINEER:
PAGE: 1/1

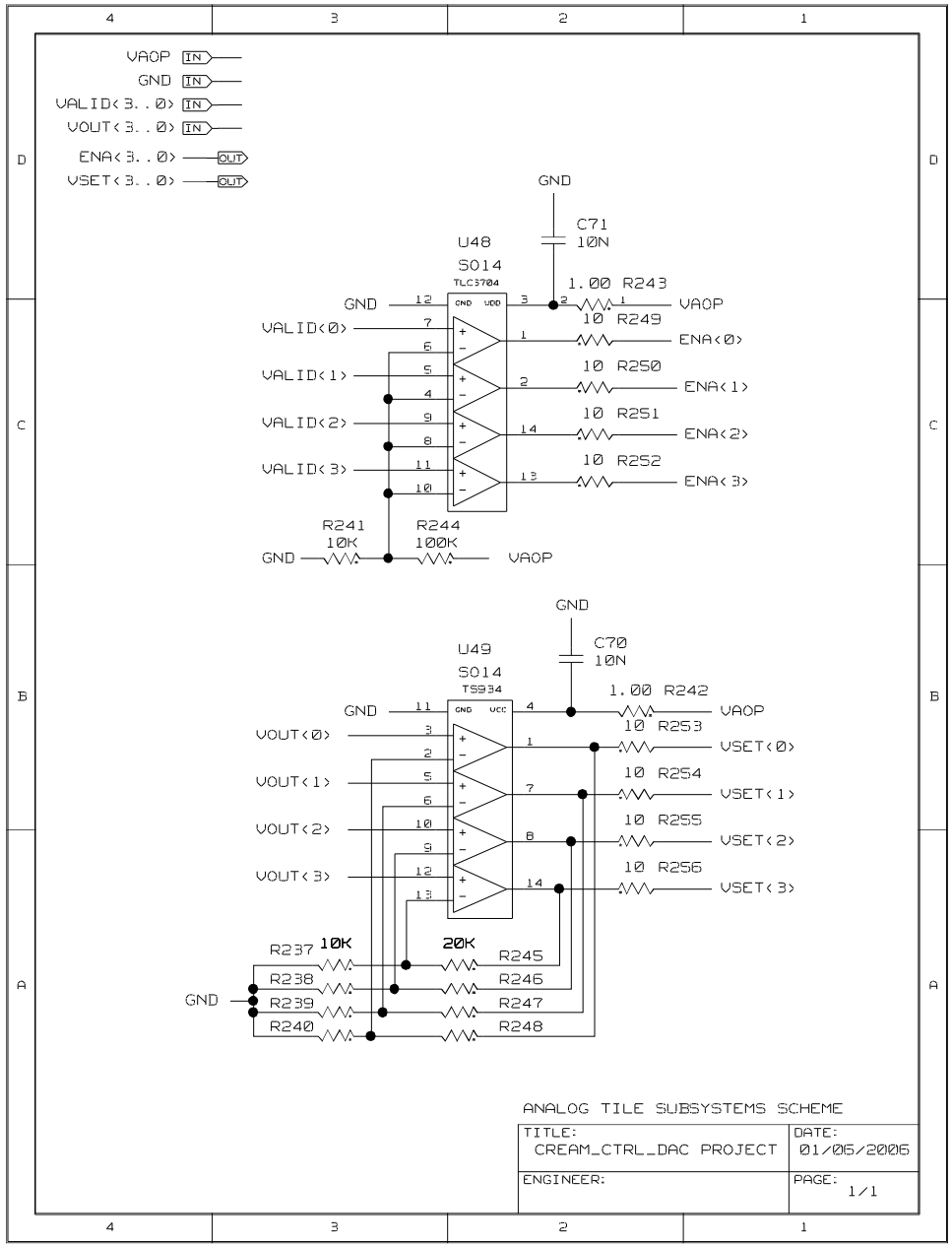

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C59
10N
U36
SO14
TLC3704
1.00 R123
VAOP
10 R129
ENA<0>
10 R130
ENA<1>
10 R131
ENA<2>
10 R132
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R121
10K
R124
100K
U37
SO14
TS934
C58
10N
1.00 R122
10 R133
VSET<0>
10 R134
VSET<1>
10 R135
VSET<2>
10 R136
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R117 10K
20K
R125
R118
R126
R119
R127
R120
R128
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

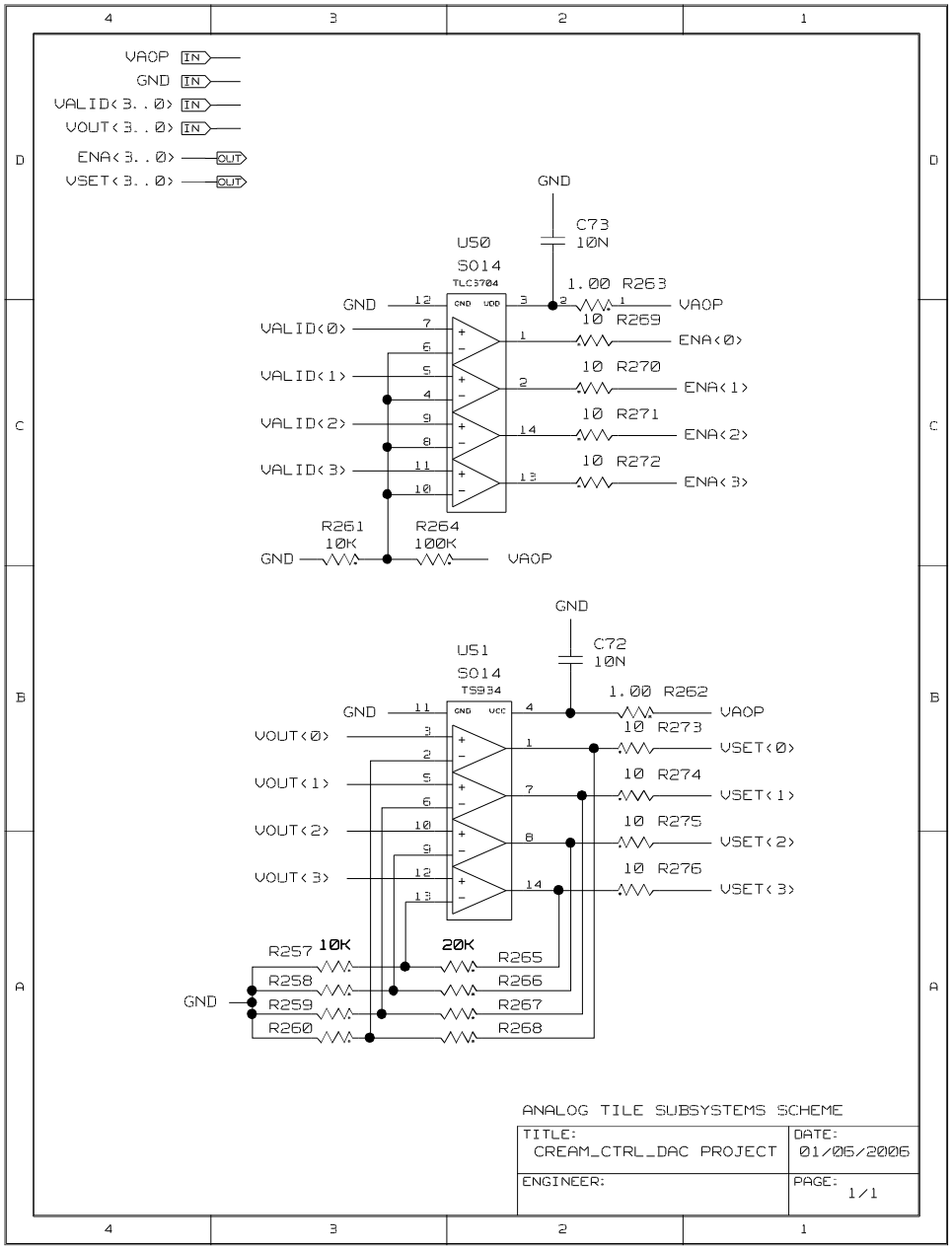

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C61
10N
U38
SO14
TLC3704
1.00 R143
VAOP
10 R149
ENA<0>
10 R150
ENA<1>
10 R151
ENA<2>
10 R152
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R141
10K
R144
100K
U39
SO14
TS934
C60
10N
1.00 R142
10 R153
VSET<0>
10 R154
VSET<1>
10 R155
VSET<2>
10 R156
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R137 10K
20K
R145
R138
R146
R139
R147
R140
R148
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

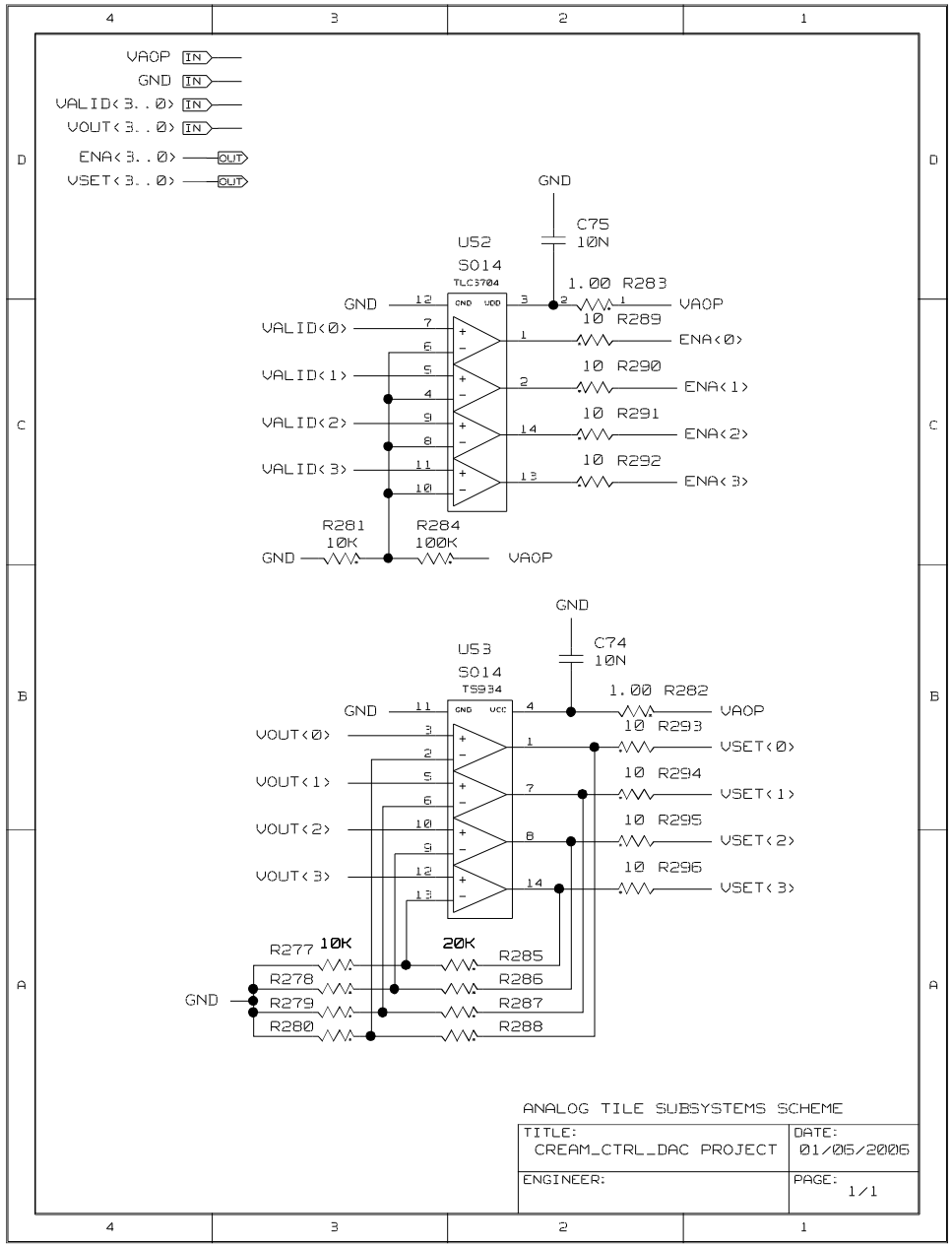

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C65
10N
U42
S014
TLC3704
1.00 R183
VAOP
10 R189
ENA<0>
10 R190
ENA<1>
10 R191
ENA<2>
10 R192
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R181
10K
R184
100K
U43
S014
TS934
C64
10N
1.00 R182
10 R193
VSET<0>
10 R194
VSET<1>
10 R195
VSET<2>
10 R196
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R177 10K
20K
R185
R178
R186
R179
R187
R180
R188
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

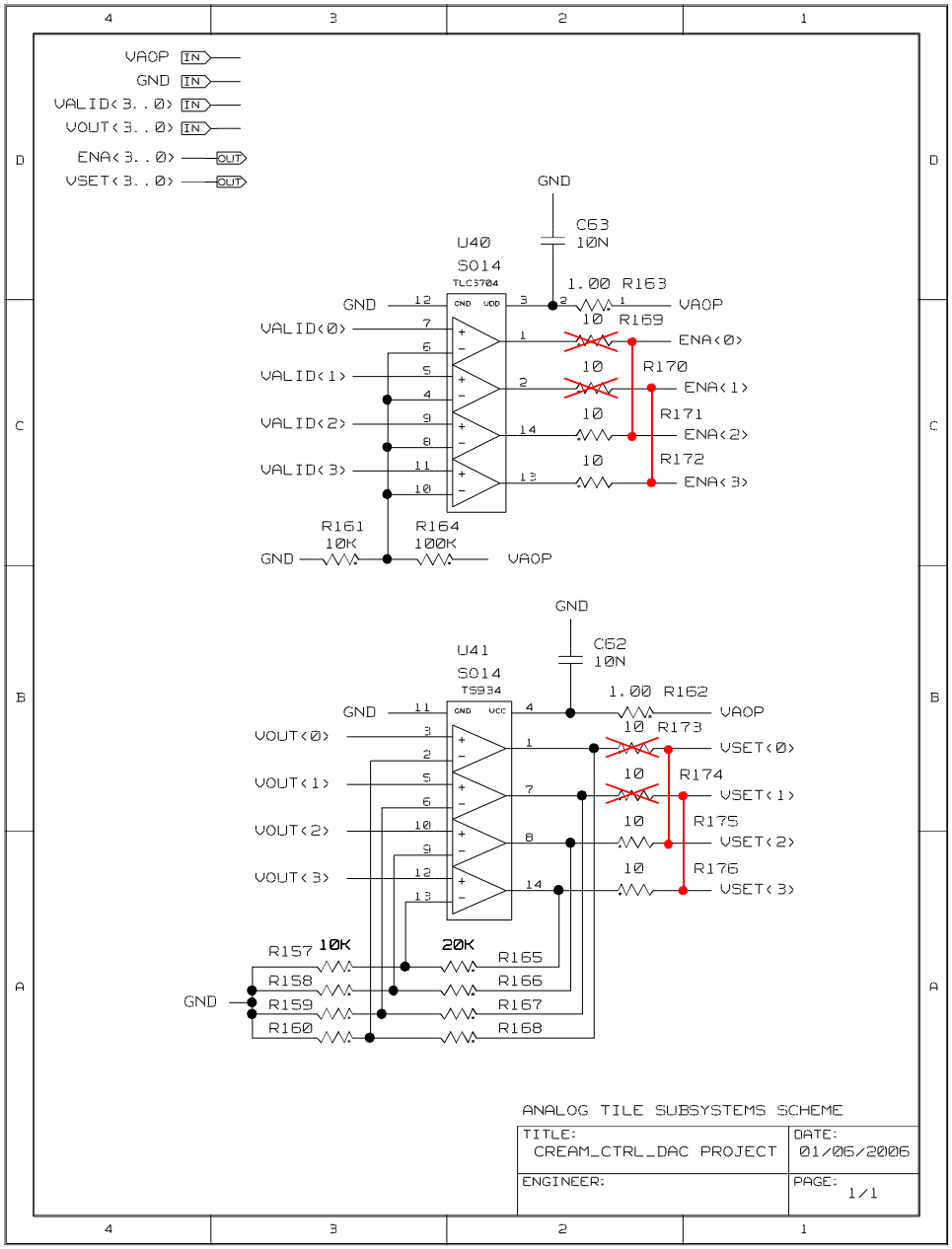

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C67
10N
U44
S014
TLC3704
1.00 R203
VAOP
10 R209
ENA<0>
10 R210
ENA<1>
10 R211
ENA<2>
10 R212
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R201
10K
R204
100K
U45
S014
TS934
C66
10N
1.00 R202
10 R213
VSET<0>
10 R214
VSET<1>
10 R215
VSET<2>
10 R216
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R197 10K
20K
R205
R198
R206
R199
R207
R200
R208
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

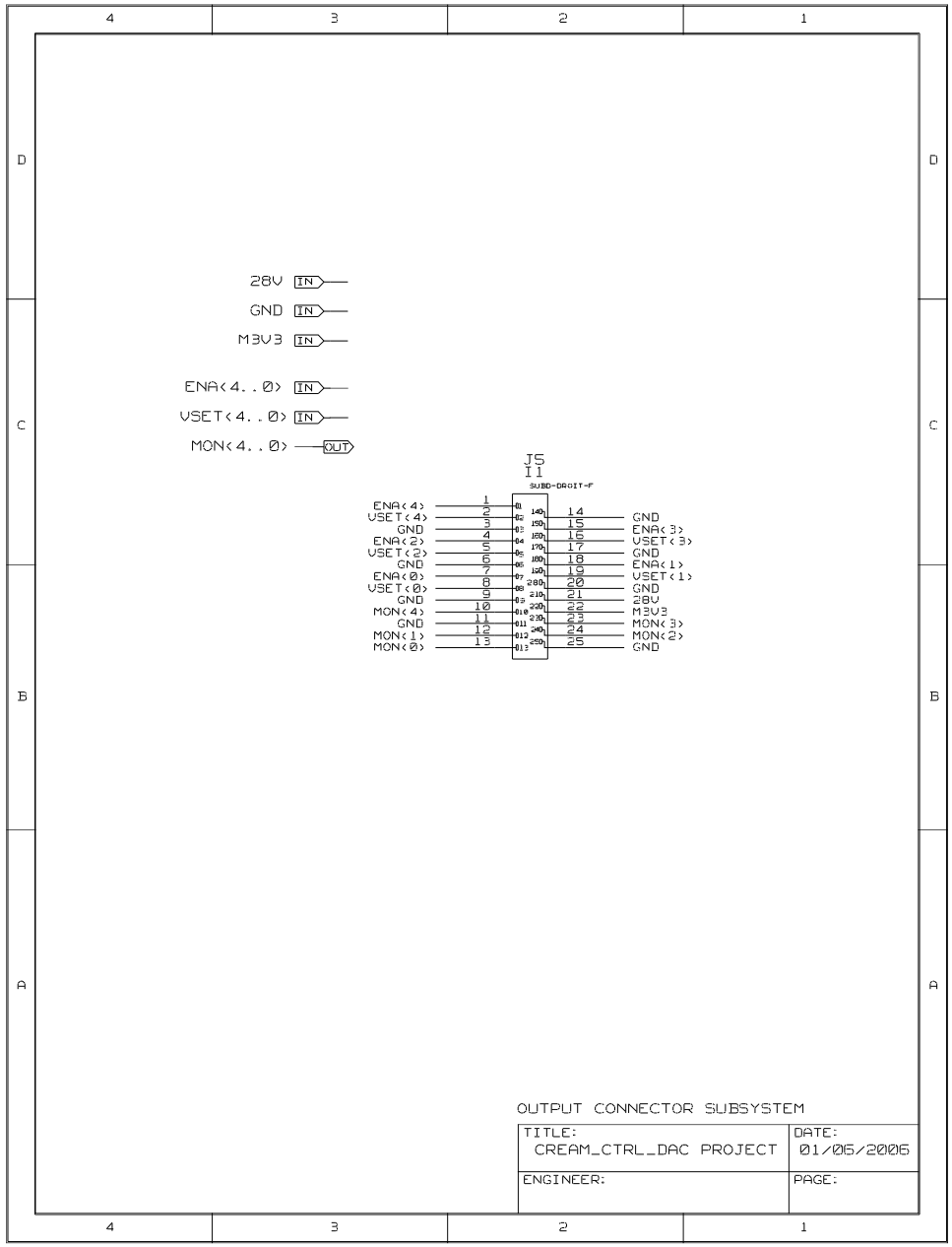

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C69
10N
U46
S014
TLC3704
1.00 R223
VAOP
10 R229
ENA<0>
10 R230
ENA<1>
10 R231
ENA<2>
10 R232
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R221
10K
R224
100K
U47
C68
TS934
1.00 R222
10 R233
VSET<0>
10 R234
VSET<1>
10 R235
VSET<2>
10 R236
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R217 10K
20K
R225
R218
R226
R219
R227
R220
R228
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

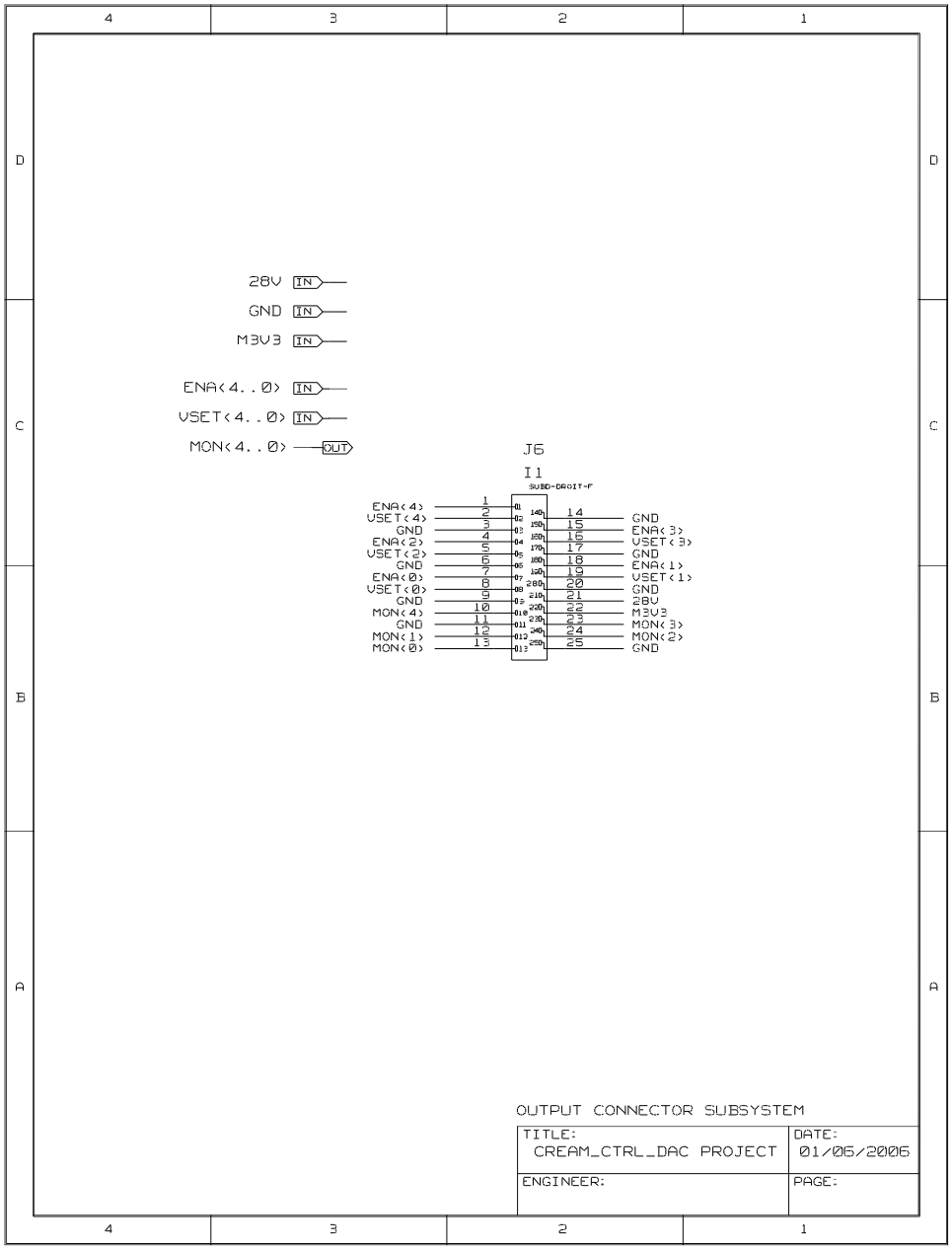

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C71
10N
U48
S014
TLC3704
1.00 R243
VAOP
10 R249
ENA<0>
10 R250
ENA<1>
10 R251
ENA<2>
10 R252
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R241
10K
R244
100K
U49
C70
TS934
1.00 R242
10 R253
VSET<0>
10 R254
VSET<1>
10 R255
VSET<2>
10 R256
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R237 10K
20K
R245
R238
R246
R239
R247
R240
R248
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

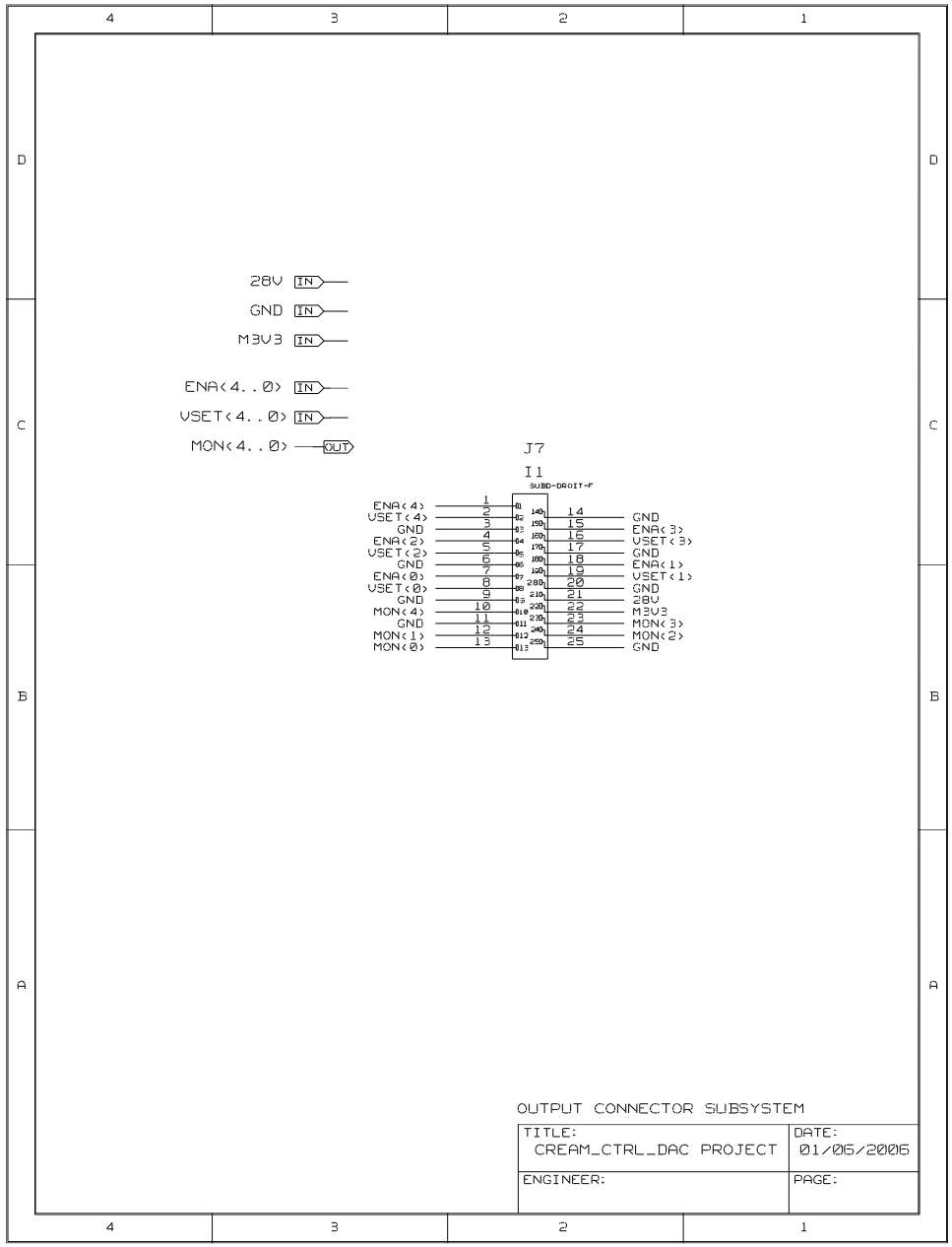

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C73
10N
U50
S014
TLC3704
1.00 R263
VAOP
10 R269
ENA<0>
10 R270
ENA<1>
10 R271
ENA<2>
10 R272
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R261
10K
R264
100K
U51
S014
TS934
C72
10N
1.00 R262
10 R273
VSET<0>
10 R274
VSET<1>
10 R275
VSET<2>
10 R276
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R257 10K
20K
R265
R258
R266
R259
R267
R260
R268
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

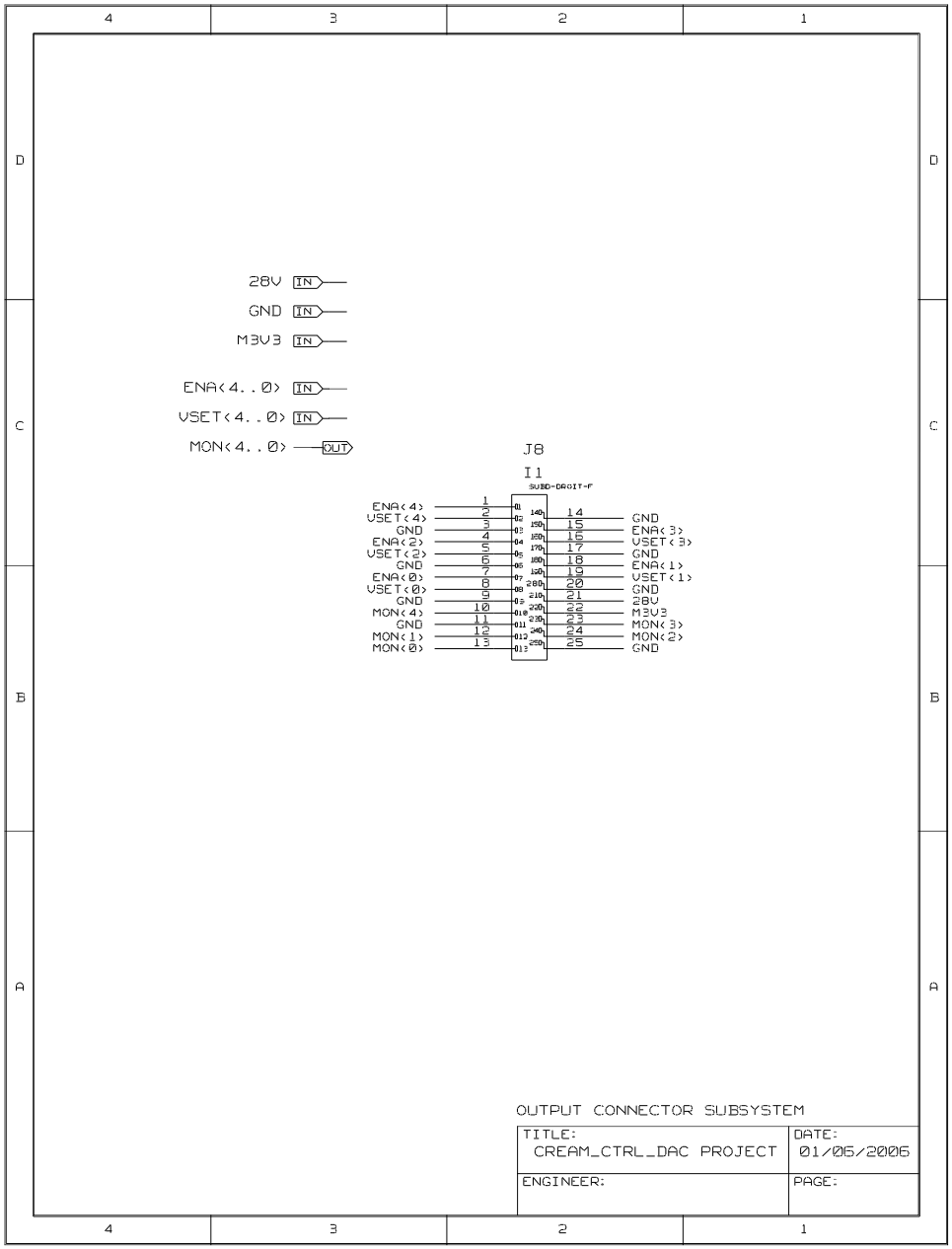

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C75
10N
U52
S014
TLC3704
1.00 R283
VAOP
10 R289
ENA<0>
10 R290
ENA<1>
10 R291
ENA<2>
10 R292
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R281
10K
R284
100K
U53
S014
TS934
C74
10N
1.00 R282
10 R293
VSET<0>
10 R294
VSET<1>
10 R295
VSET<2>
10 R296
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R277 10K
20K
R285
R278
R286
R279
R287
R280
R288
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

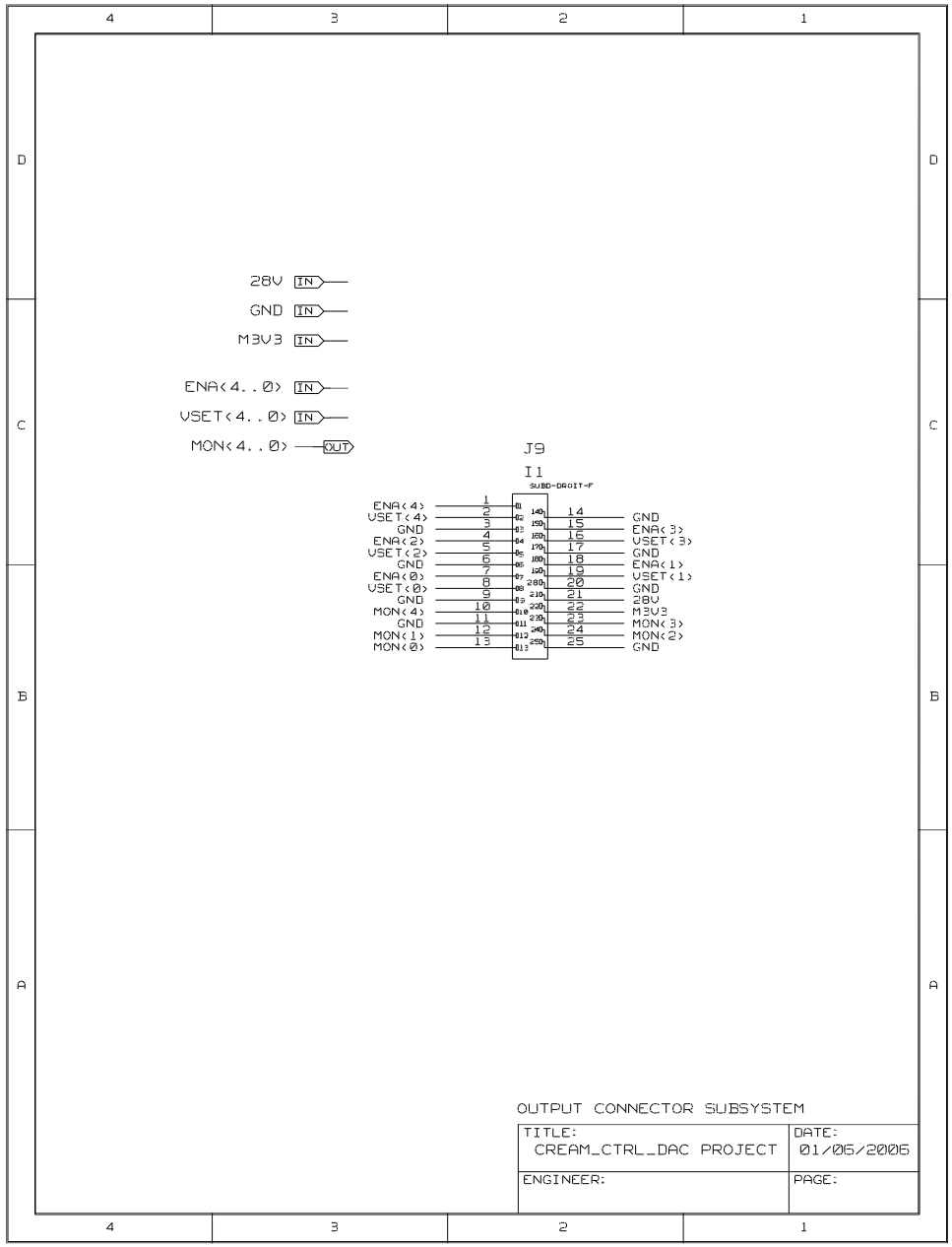

VAOP IN
GND IN
VALID<3..0> IN
VOUT<3..0> IN
ENA<3..0> OUT
VSET<3..0> OUT
GND
C63
10N
U40
S014
TLC3704
1.00 R163
VAOP
10 R169
ENA<0>
10 R170
ENA<1>
10 R171
ENA<2>
10 R172
ENA<3>
VALID<0>
VALID<1>
VALID<2>
VALID<3>
R161
10K
R164
100K
U41
S014
TS934
C62
10N
1.00 R162
10 R173
VSET<0>
10 R174
VSET<1>
10 R175
VSET<2>
10 R176
VSET<3>
VOUT<0>
VOUT<1>
VOUT<2>
VOUT<3>
R157 10K
20K
R165
R158
R166
R159
R167
R160
R168
ANALOG TILE SUBSYSTEMS SCHEME
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE: 1/1

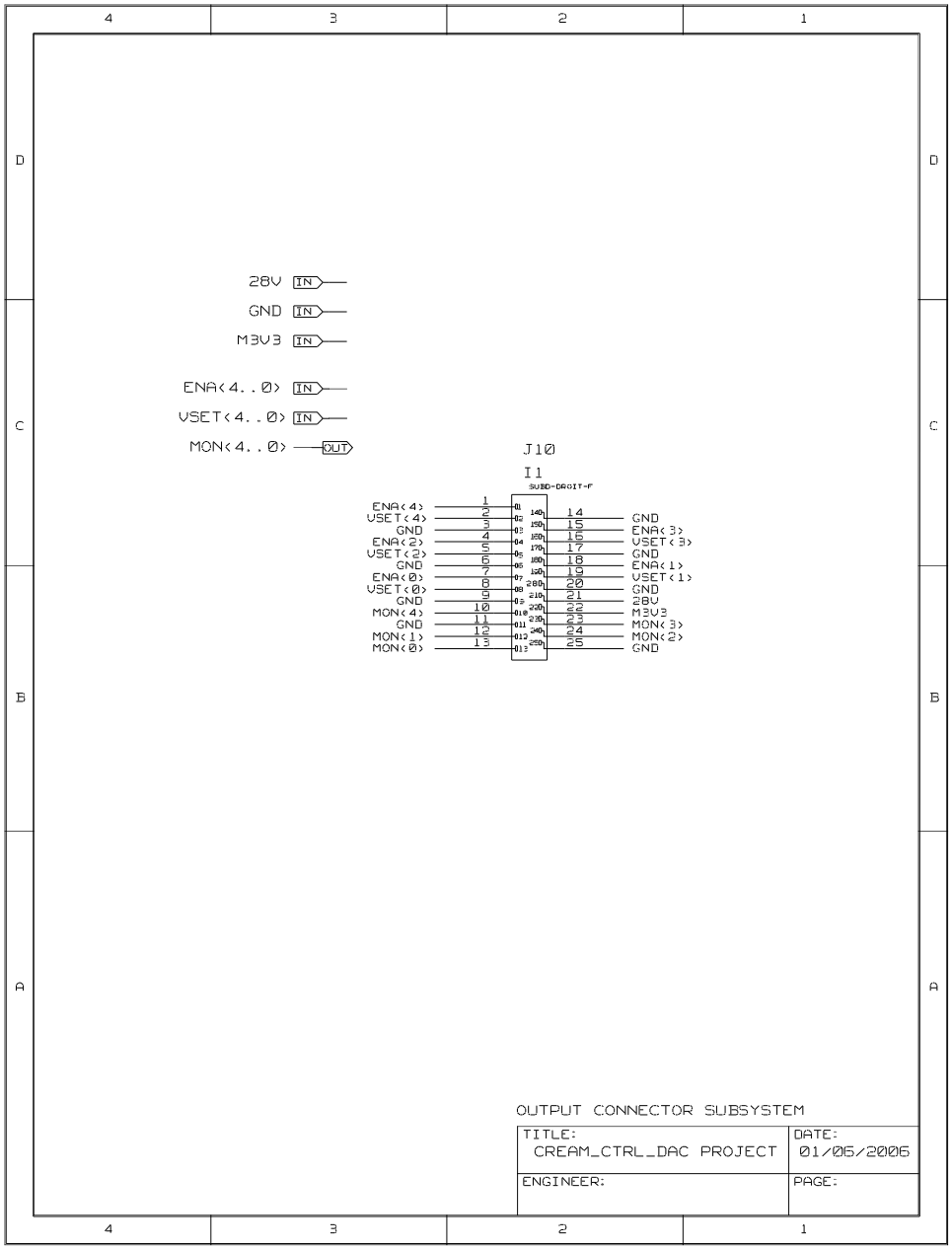

28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J5
I1
SUBD-DRGIT-F
ENA<4>
VSET<4>
GND
ENA<2>
VSET<2>
GND
ENA<0>
VSET<0>
GND
MON<4>
GND
MON<1>
MON<0>
GND
GND
ENA<3>
VSET<3>
GND
ENA<1>
VSET<1>
GND
28V
M3V3
MON<3>
MON<2>
GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE:

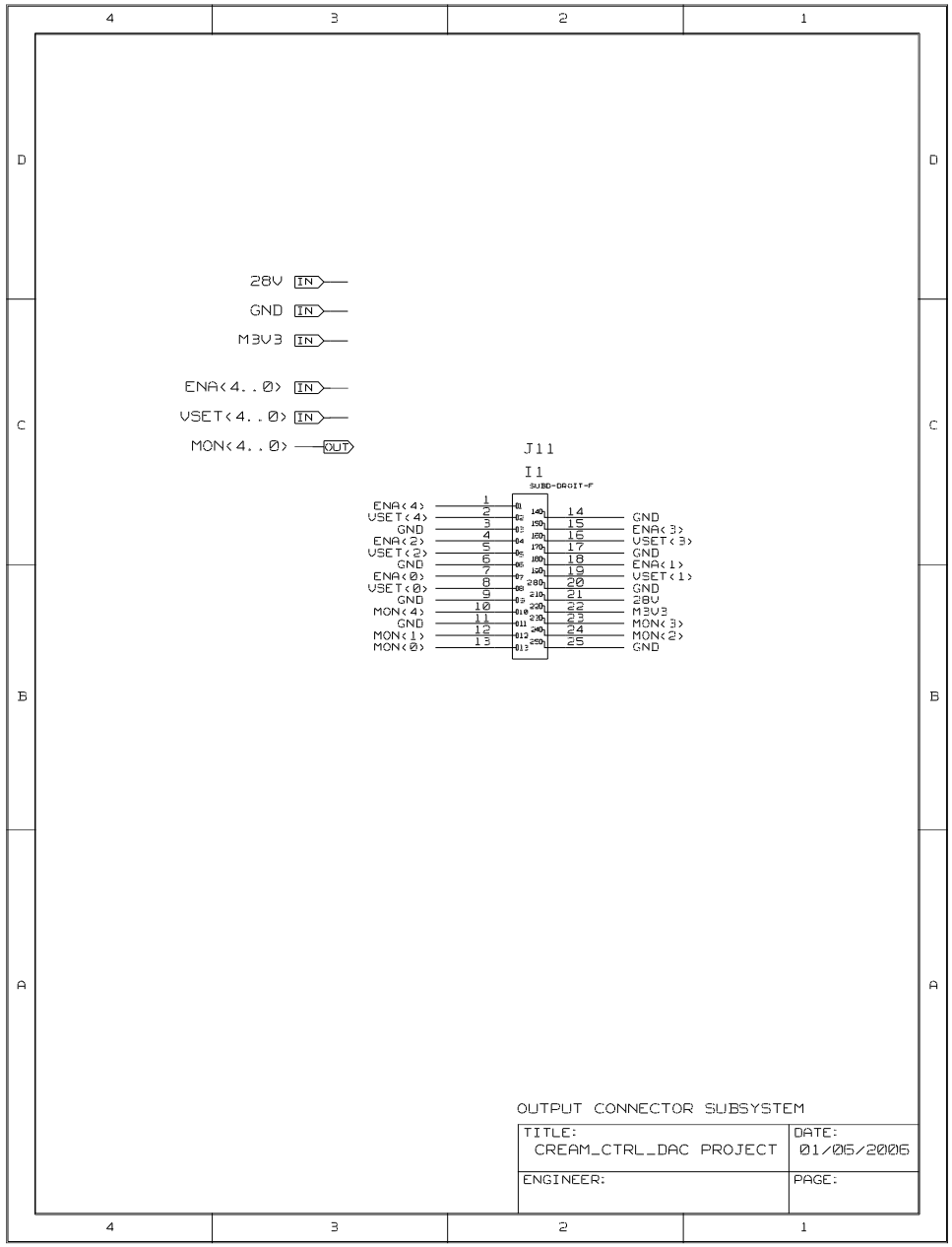

28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J6
I1
SUBD-DAQIT-F
ENA<4> 1
VSET<4> 2
GND 3
ENA<2> 4
VSET<2> 5
GND 6
ENA<0> 7
VSET<0> 8
GND 9
MON<4> 10
GND 11
MON<1> 12
MON<0> 13
14 GND
15 ENA<3>
16 VSET<3>
17 GND
18 ENA<1>
19 VSET<1>
20 GND
21 28V
22 M3V3
23 MON<3>
24 MON<2>
25 GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE:

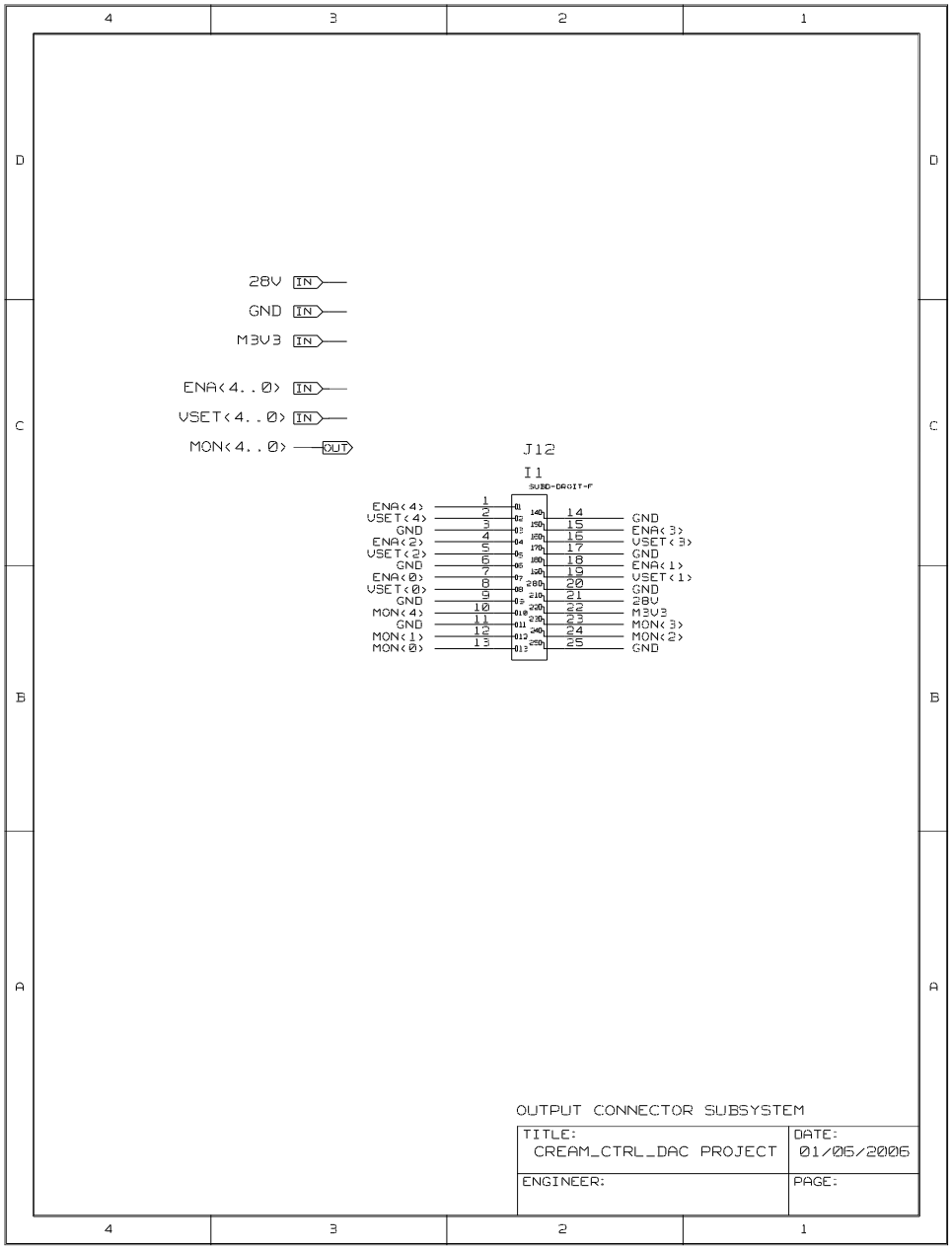

28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J7
I1
SUBD-DAQIT-F
ENA<4> 1
VSET<4> 2
GND 3
ENA<2> 4
VSET<2> 5
GND 6
ENA<0> 7
VSET<0> 8
GND 9
MON<4> 10
GND 11
MON<1> 12
MON<0> 13
14 GND
15 ENA<3>
16 VSET<3>
17 GND
18 ENA<1>
19 VSET<1>
20 GND
21 28V
22 M3V3
23 MON<3>
24 MON<2>
25 GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE:

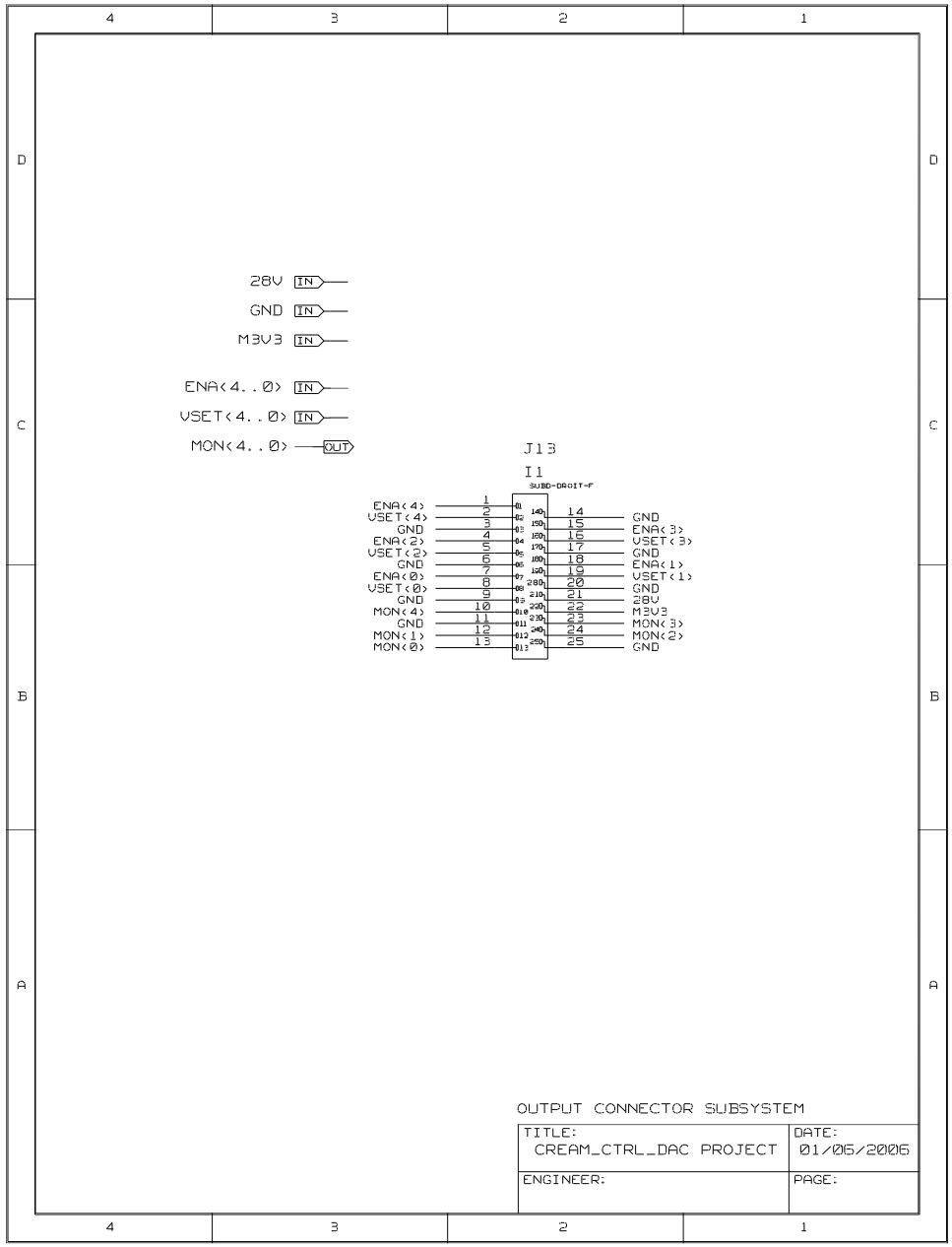

28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J8
I1
SUBD-DAQIT-F
ENA<4> 1
VSET<4> 2
GND 3
ENA<2> 4
VSET<2> 5
GND 6
ENA<0> 7
VSET<0> 8
GND 9
MON<4> 10
GND 11
MON<1> 12
MON<0> 13
14 GND
15 ENA<3>
16 VSET<3>
17 GND
18 ENA<1>
19 VSET<1>
20 GND
21 28V
22 M3V3
23 MON<3>
24 MON<2>
25 GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE:

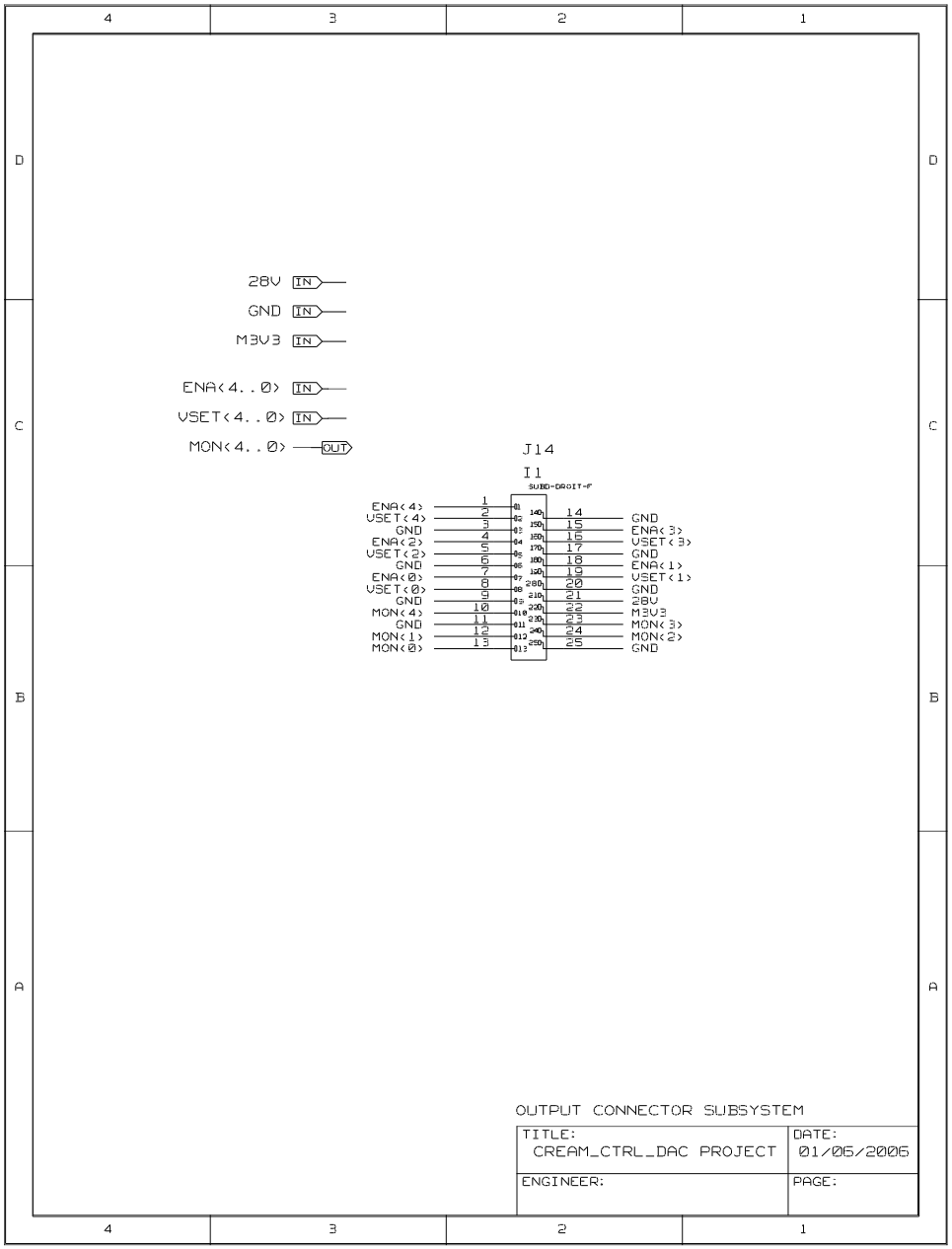

28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J9
I1
SUBD-DAQIT-F
ENA<4> 1
VSET<4> 2
GND 3
ENA<2> 4
VSET<2> 5
GND 6
ENA<0> 7
VSET<0> 8
GND 9
MON<4> 10
GND 11
MON<1> 12
MON<0> 13
14 GND
15 ENA<3>
16 VSET<3>
17 GND
18 ENA<1>
19 VSET<1>
20 GND
21 28V
22 M3V3
23 MON<3>
24 MON<2>
25 GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/06/2006
ENGINEER:
PAGE:

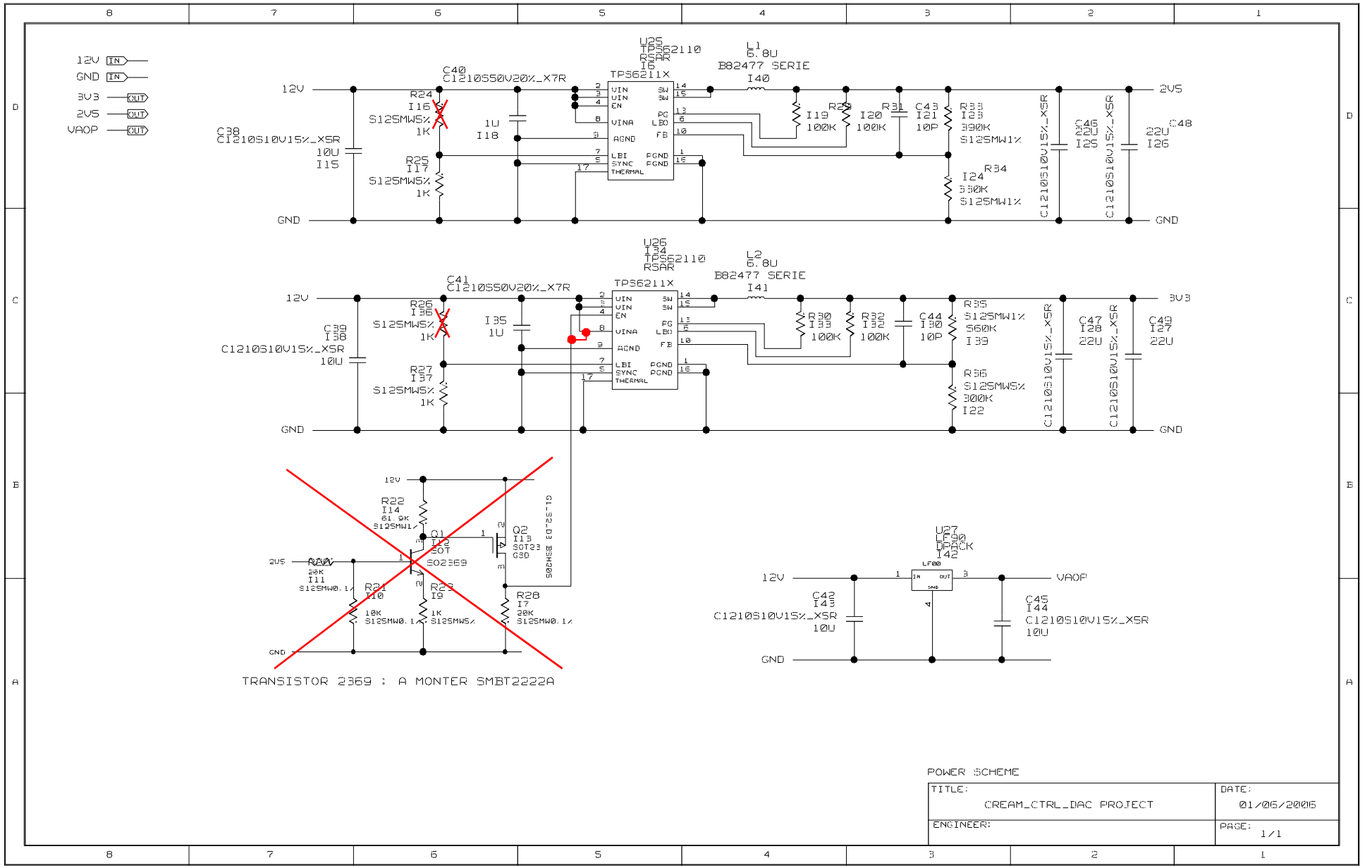

28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J10
I1
SUBD-DRGIT-F
ENA<4> 1
VSET<4> 2
GND 3
ENA<2> 4
VSET<2> 5
GND 6
ENA<0> 7
VSET<0> 8
GND 9
MON<4> 10
GND 11
MON<1> 12
MON<0> 13
14 GND
15 ENA<3>
16 VSET<3>
17 GND
18 ENA<1>
19 VSET<1>
20 GND
21 28V
22 M3V3
23 MON<3>
24 MON<2>
25 GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2006
ENGINEER:
PAGE:


28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J11
I1
SUBD-DRGIT-F
ENA<4> 1
VSET<4> 2
GND 3
ENA<2> 4
VSET<2> 5
GND 6
ENA<0> 7
VSET<0> 8
GND 9
MON<4> 10
GND 11
MON<1> 12
MON<0> 13
14 GND
15 ENA<3>
16 VSET<3>
17 GND
18 ENA<1>
19 VSET<1>
20 GND
21 28V
22 M3V3
23 MON<3>
24 MON<2>
25 GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2006
ENGINEER:
PAGE:

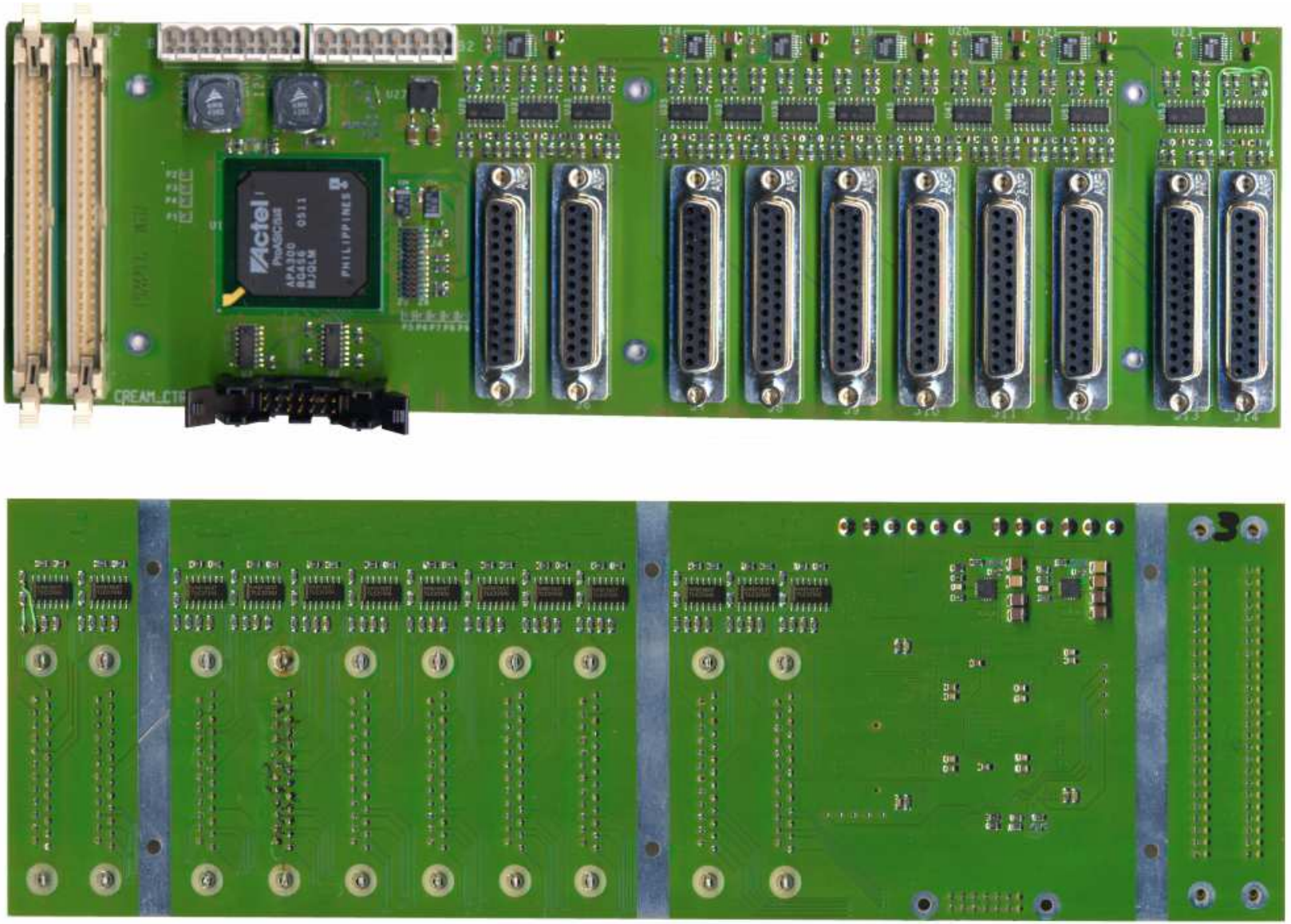

28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J12
I1
SUBD-DRGIT-F
ENA<4> 1
VSET<4> 2
GND 3
ENA<2> 4
VSET<2> 5
GND 6
ENA<0> 7
VSET<0> 8
GND 9
MON<4> 10
GND 11
MON<1> 12
MON<0> 13
14 GND
15 ENA<3>
16 VSET<3>
17 GND
18 ENA<1>
19 VSET<1>
20 GND
21 28V
22 M3V3
23 MON<3>
24 MON<2>
25 GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2006
ENGINEER:
PAGE:


28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J13
I1
SUBD-DRGIT-F
ENA<4> 1
VSET<4> 2
GND 3
ENA<2> 4
VSET<2> 5
GND 6
ENA<0> 7
VSET<0> 8
GND 9
MON<4> 10
GND 11
MON<1> 12
MON<0> 13
14 GND
15 ENA<3>
16 VSET<3>
17 GND
18 ENA<1>
19 VSET<1>
20 GND
21 28V
22 M3V3
23 MON<3>
24 MON<2>
25 GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE: CREAM_CTRL_DAC PROJECT
DATE: 01/05/2006
ENGINEER:
PAGE:

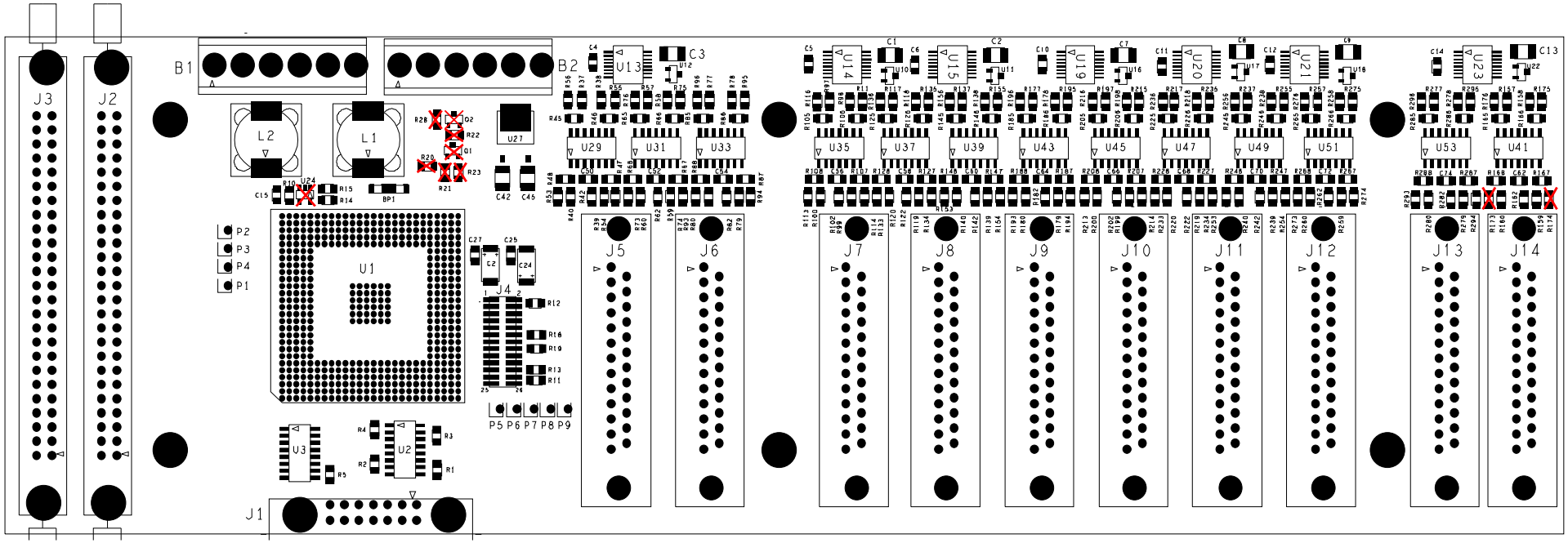

28V IN
GND IN
M3V3 IN
ENA<4..0> IN
VSET<4..0> IN
MON<4..0> OUT
J14
I1
SUBD-DROIT-F
ENA<4>
VSET<4>
GND
ENA<2>
VSET<2>
GND
ENA<0>
VSET<0>
GND
MON<4>
GND
MON<1>
MON<0>
GND
ENA<3>
VSET<3>
GND
ENA<1>
VSET<1>
GND
28V
M3V3
MON<3>
MON<2>
GND
OUTPUT CONNECTOR SUBSYSTEM
TITLE:
CREAM_CTRL_DAC PROJECT
DATE:
01/05/2005
ENGINEER:
PAGE:


12V IN
GND IN
3V3 OUT
2V5 OUT
VAOP OUT
U25
TPS62110
RSAR
I6
L1
6.8U
B82477 SERIE
I40
C40
C1210S50V20%_X7R
TPS6211X
12V
GND
R24
I16
S125MW5%
1K
R25
I17
S125MW5%
1K
C38
C1210S10V15%_XSR
10U
I15
1U
I18
VIN
VIN
EN
VINA
AGND
LBI
SYNC
THERMAL
SW
SW
PG
LBO
FB
PGND
PGND
R29
I19
100K
R28
I20
100K
R31
C43
I21
10P
R33
I38
390K
S125MW1%
R34
I24
330K
S125MW1%
C46
I25
22U
C1210S10V15%_XSR
C48
22U
I26
C1210S10V15%_XSR
2V5
GND
U26
I34
TPS62110
RSAR
L2
6.8U
B82477 SERIE
I41
C41
C1210S50V20%_X7R
TPS6211X
12V
GND
R26
I36
S125MW5%
1K
R27
I37
S125MW5%
1K
C39
I39
C1210S10V15%_XSR
10U
I35
1U
R30
I33
100K
R32
I32
100K
C44
I30
10P
R35
S125MW1%
560K
I39
R36
S125MW5%
300K
I22
C1210S10V15%_XSR
C47
I28
22U
C49
I27
22U
C1210S10V15%_XSR
3V3
GND
12V
R22
I14
61.9K
S125MW1%
Q1
I12
SOT
SO2369
Q2
I13
SOT23
G3D
BSH205
2V5
R20
20K
I11
S125MW0.1%
R21
I10
10K
S125MW0.1%
R23
I9
1K
S125MW5%
R28
I7
20K
S125MW0.1%
GND
TRANSISTOR 2369 : A MONTER SMBT2222A
U27
LF50
DPACK
I42
LF00
12V
VAOP
GND
C42
I43
C1210S10V15%_XSR
10U
C45
I44
C1210S10V15%_XSR
10U
POWER SCHEME
TITLE:
CREAM_CTRL_DAC PROJECT
DATE:
01/06/2006
ENGINEER:
PAGE:
1/1

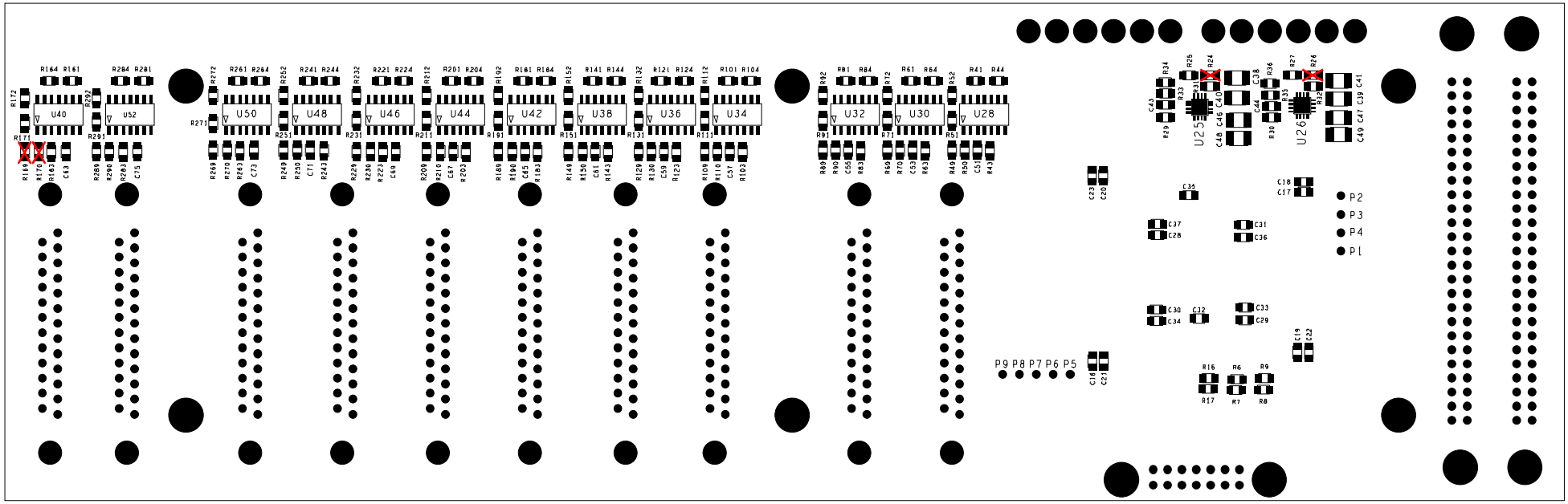
Annex 8 : Board picture



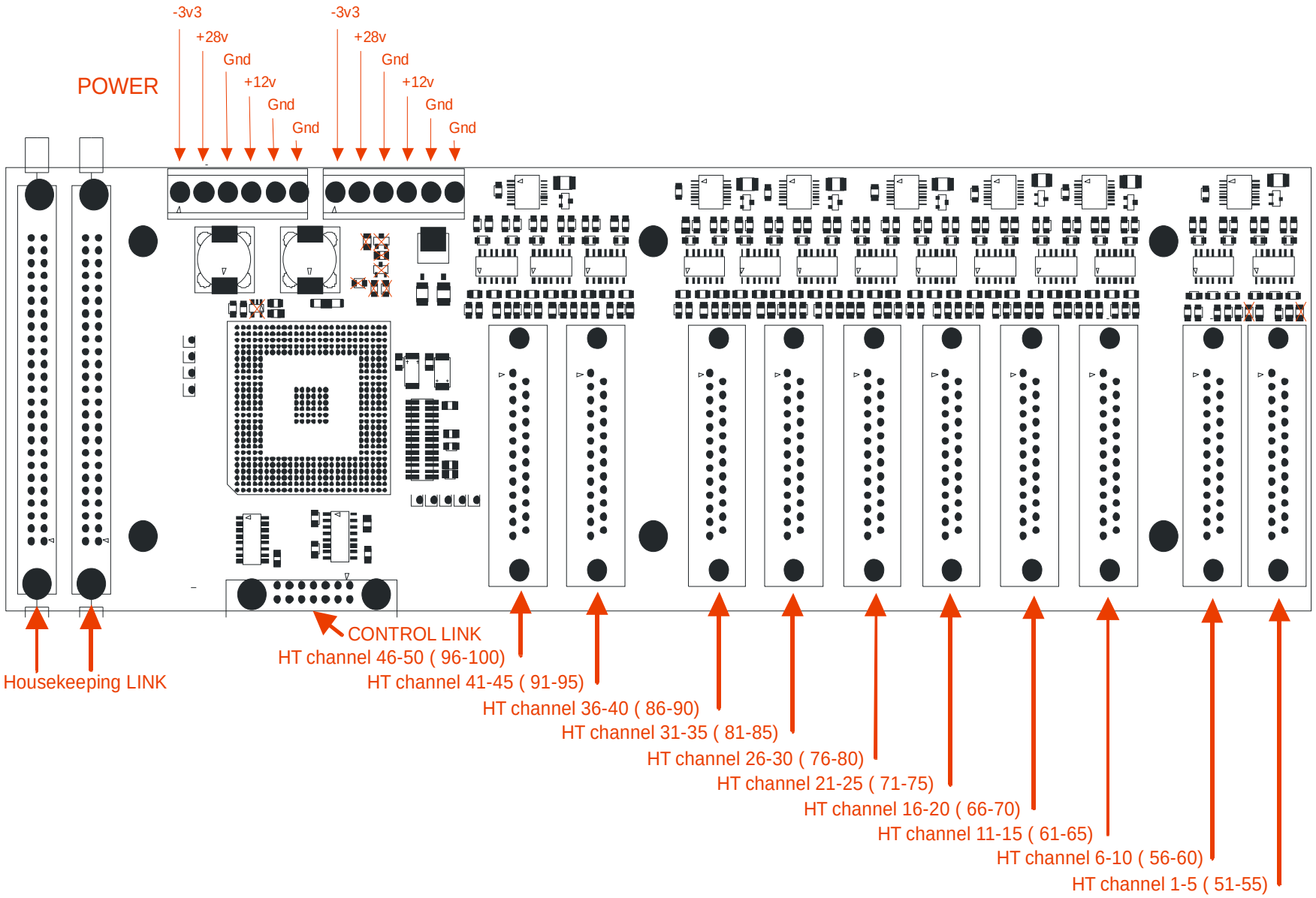
Annex 9 : Board implantation components (top side)



Annex 10 : Board implantation components (bottom side)



Annex 11 : High voltage command board connector designation



Annex 12 : FPGA VHDL file description

```
--=============================================================================
--
--  Design Units : CREAM HT control board, CREAM experiment
--
--  File name    : cdac.vhd
--
--  Purpose      : This module describe the FPGA who make the interface
--                 between the CREAM computer Unit and the Photo
--                 Multiplicator high voltage units. This FPGA generate :
--                   * a serial stream to the DAC components ( 7 ) with
--                     there all control signal ( Chip select and load
--                     signals )
--                   * a logical validation signal for each High voltage
--                     Units ( 50 )
--
--  Notes        : Synthesize with Synplify lite for Actel 7.7.1B
--                 Placed with Designer software 6.1.1
--                 Libero ide v6.1 SP1
--                 Simulated with Modelsim Actel 5.8d
--
--  Limitations  :
--
--  Errors       :
--
--  Library      :
--
--  Dependencies :
--
--  Author       : Olivier TRAORE, Joel BOUVIER
--                 Laboratoire de physique Subatomique et de cosmologie
--                 53 Avenue des Martyrs
--                 38026 Grenoble Cedex, FRANCE
--
--
=============================================================================
-- Revision List
-- Version  Author  Date      Change
-- 0.0    O. TRAORE 25/07/05  Initial version
--  1        JB     25/10/06  add Test point Output
--  2        JB     26/11/06  add a table for the rerouting of the channel
--                             internal address to meet for an easier
--                             interconnect with the HT module
--
=============================================================================
==

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity cdac is
  port(
    LCLK      : in  std_logic;
    CLOCK_IN  : in  std_logic;
    CLOCK_OUT : out std_logic;

-----------------------------------------------
-- INPUT SIGNAL FROM RS422 TO LVTTL TRANSLATOR
-- Interface with the Computer Unit
--
--  nMODE = 0 => 0 < @ < 50, nMODE = 1 => 0 < @ < 100
--  nFIRST active if nMODE = 1
--  nFIRST = 0 => 0 < @ < 50
--  nFIRST = 1 => 51 < @ < 100
-----------------------------------------------

    nRESET    : in  std_logic;
    CLOCK     : in  std_logic;
    DATA      : in  std_logic;
    ADDREN    : in  std_logic;
    DATAEN    : in  std_logic;
    nMODE     : in  std_logic;
    nFIRST    : in  std_logic;

-- OUTPUT SIGNAL FROM ACTEL CPLD TO DAC COMPONENT
-----------------------------------------------

    dac_ck    : out std_logic_vector(6 downto 0);
    NCS       : out std_logic_vector(6 downto 0);
    DATA_OUT  : out std_logic_vector(6 downto 0);

-- OUTPUT SIGNAL FROM ACTEL CPLD FOR VALIDATION

   VALID      : out std_logic_vector(49 downto 0);

-- Test points

   test       : out std_logic_vector( 9 downto 1) );
end cdac;

architecture RTL of cdac is

-- ADDRESS – COMMAND – COUNTER – REGISTER

signal address_send      : std_logic_vector(7 downto 0);

signal address_shifter   : std_logic_vector(7 downto 0);
signal CS1               : std_logic_vector(6 downto 0);
signal address_valid     : std_logic;

-- COMPARATORS OUTPUT SIGNAL “ 0XAA – 0X0A – 0X05 “

signal COMP_ENABLE     : std_logic;
signal COMP_DISABLE    : std_logic;

-- DAC MODULE OUTPUTS GENERATION

type DEF_STATE is (IDDLE, COMMAND, VERIF, CMD_ACK, DAC_VAL0, DAC_VAL1,
                   DAC_INH0, DAC_INH1, DAC_DAT0, DAC_DAT1, DAC_DAT2,
                   DAC_DAT3, DAC_DAT4, DAC_DAT5, DAC_DAT6,FINISH);
signal STATE       : DEF_STATE;
signal reg_data    : std_logic_vector(7 downto 0);
```

```
signal ENABLE_SYNC : std_logic;
signal count_seq   : std_logic_vector(3 downto 0);
signal DATA_INT    : std_logic;

-- VALIDATION COMMAND LINES FROM 0 TO 49

signal CSVALID : std_logic_vector(49 downto 0);

begin

CLOCK_OUT <= CLOCK_IN ;
dac_ck    <= CLOCK_IN & CLOCK_IN & CLOCK_IN & CLOCK_IN
           & CLOCK_IN & CLOCK_IN & CLOCK_IN ;

-- ADDRESS REGISTER PROCESS

ADDRESS_PROCESS : process(nRESET,CLOCK)
begin
  if nRESET ='0' then                        -- Low level active RESET
    address_send <= (others=>'0');
  elsif rising_edge(CLOCK) then
    if ADDREN = '0' then
       address_send(7 downto 0) <= address_send(6 downto 0) & DATA ;
    end if ;
  end if ;
end process ADDRESS_PROCESS ;

-- ASSERT CHIP SELECT WITH ADDRESS VALUE

process ( nRESET, CLOCK )
begin
  if nRESET ='0' then
    CS1              <= ( others => '1');
    address_valid    <= '0';
    address_shifter  <= ( others => '1');
  elsif rising_edge(CLOCK) then

    case address_send is
      when x"01"  => address_shifter <= conv_std_logic_vector(45,8); -- 1
      when x"02"  => address_shifter <= conv_std_logic_vector(46,8); -- 2
      when x"03"  => address_shifter <= conv_std_logic_vector(47,8); -- 3
      when x"04"  => address_shifter <= conv_std_logic_vector(48,8); -- 4
      when x"05"  => address_shifter <= conv_std_logic_vector(49,8); -- 5
      when x"06"  => address_shifter <= conv_std_logic_vector(40,8); -- 6
      when x"07"  => address_shifter <= conv_std_logic_vector(41,8); -- 7
      when x"08"  => address_shifter <= conv_std_logic_vector(42,8); -- 8
      when x"09"  => address_shifter <= conv_std_logic_vector(43,8); -- 9
      when x"0A"  => address_shifter <= conv_std_logic_vector(44,8); -- 10
      when x"0B"  => address_shifter <= conv_std_logic_vector(35,8); -- 11
      when x"0C"  => address_shifter <= conv_std_logic_vector(36,8); -- 12
      when x"0D"  => address_shifter <= conv_std_logic_vector(37,8); -- 13
      when x"0E"  => address_shifter <= conv_std_logic_vector(38,8); -- 14
      when x"0F"  => address_shifter <= conv_std_logic_vector(39,8); -- 15
      when x"10"  => address_shifter <= conv_std_logic_vector(30,8); -- 16
      when x"11"  => address_shifter <= conv_std_logic_vector(31,8); -- 17
      when x"12"  => address_shifter <= conv_std_logic_vector(32,8); -- 18
      when x"13"  => address_shifter <= conv_std_logic_vector(33,8); -- 19
      when x"14"  => address_shifter <= conv_std_logic_vector(34,8); -- 20
      when x"15"  => address_shifter <= conv_std_logic_vector(25,8); -- 21
      when x"16"  => address_shifter <= conv_std_logic_vector(26,8); -- 22
      when x"17"  => address_shifter <= conv_std_logic_vector(27,8); -- 23
      when x"18"  => address_shifter <= conv_std_logic_vector(28,8); -- 24
      when x"19"  => address_shifter <= conv_std_logic_vector(29,8); -- 25
      when x"1A"  => address_shifter <= conv_std_logic_vector(20,8); -- 26
      when x"1B"  => address_shifter <= conv_std_logic_vector(21,8); -- 27
      when x"1C"  => address_shifter <= conv_std_logic_vector(22,8); -- 28
      when x"1D"  => address_shifter <= conv_std_logic_vector(23,8); -- 29
      when x"1E"  => address_shifter <= conv_std_logic_vector(24,8); -- 30
      when x"1F"  => address_shifter <= conv_std_logic_vector(15,8); -- 31
      when x"20"  => address_shifter <= conv_std_logic_vector(16,8); -- 32
      when x"21"  => address_shifter <= conv_std_logic_vector(17,8); -- 33
      when x"22"  => address_shifter <= conv_std_logic_vector(18,8); -- 34
      when x"23"  => address_shifter <= conv_std_logic_vector(19,8); -- 35
      when x"24"  => address_shifter <= conv_std_logic_vector(10,8); -- 36
      when x"25"  => address_shifter <= conv_std_logic_vector(11,8); -- 37
      when x"26"  => address_shifter <= conv_std_logic_vector(12,8); -- 38
      when x"27"  => address_shifter <= conv_std_logic_vector(13,8); -- 39
      when x"28"  => address_shifter <= conv_std_logic_vector(14,8); -- 40
      when x"29"  => address_shifter <= conv_std_logic_vector(5,8);  -- 41
      when x"2A"  => address_shifter <= conv_std_logic_vector(6,8);  -- 42
      when x"2B"  => address_shifter <= conv_std_logic_vector(7,8);  -- 43
      when x"2C"  => address_shifter <= conv_std_logic_vector(8,8);  -- 44
      when x"2D"  => address_shifter <= conv_std_logic_vector(9,8);  -- 45
      when x"2E"  => address_shifter <= conv_std_logic_vector(0,8);  -- 46
      when x"2F"  => address_shifter <= conv_std_logic_vector(1,8);  -- 47
      when x"30"  => address_shifter <= conv_std_logic_vector(2,8);  -- 48
      when x"31"  => address_shifter <= conv_std_logic_vector(3,8);  -- 49
      when x"32"  => address_shifter <= conv_std_logic_vector(4,8);  -- 50
      when x"33"  => address_shifter <= conv_std_logic_vector(45,8); -- 51
      when x"34"  => address_shifter <= conv_std_logic_vector(46,8); -- 52
      when x"35"  => address_shifter <= conv_std_logic_vector(47,8); -- 53
      when x"36"  => address_shifter <= conv_std_logic_vector(48,8); -- 54
      when x"37"  => address_shifter <= conv_std_logic_vector(49,8); -- 55
      when x"38"  => address_shifter <= conv_std_logic_vector(40,8); -- 56
      when x"39"  => address_shifter <= conv_std_logic_vector(41,8); -- 57
      when x"3A"  => address_shifter <= conv_std_logic_vector(42,8); -- 58
      when x"3B"  => address_shifter <= conv_std_logic_vector(43,8); -- 59
      when x"3C"  => address_shifter <= conv_std_logic_vector(44,8); -- 60
      when x"3D"  => address_shifter <= conv_std_logic_vector(35,8); -- 61
      when x"3E"  => address_shifter <= conv_std_logic_vector(36,8); -- 62
      when x"3F"  => address_shifter <= conv_std_logic_vector(37,8); -- 63
      when x"40"  => address_shifter <= conv_std_logic_vector(38,8); -- 64
      when x"41"  => address_shifter <= conv_std_logic_vector(39,8); -- 65
      when x"42"  => address_shifter <= conv_std_logic_vector(30,8); -- 66
      when x"43"  => address_shifter <= conv_std_logic_vector(31,8); -- 67
      when x"44"  => address_shifter <= conv_std_logic_vector(32,8); -- 68
      when x"45"  => address_shifter <= conv_std_logic_vector(33,8); -- 69
      when x"46"  => address_shifter <= conv_std_logic_vector(34,8); -- 70
      when x"47"  => address_shifter <= conv_std_logic_vector(25,8); -- 71
      when x"48"  => address_shifter <= conv_std_logic_vector(26,8); -- 72
      when x"49"  => address_shifter <= conv_std_logic_vector(27,8); -- 73
      when x"4A"  => address_shifter <= conv_std_logic_vector(28,8); -- 74
      when x"4B"  => address_shifter <= conv_std_logic_vector(29,8); -- 75
      when x"4C"  => address_shifter <= conv_std_logic_vector(20,8); -- 76
      when x"4D"  => address_shifter <= conv_std_logic_vector(21,8); -- 77
      when x"4E"  => address_shifter <= conv_std_logic_vector(22,8); -- 78
      when x"4F"  => address_shifter <= conv_std_logic_vector(23,8); -- 79
      when x"50"  => address_shifter <= conv_std_logic_vector(24,8); -- 80
```

```
      when x"51"  => address_shifter <= conv_std_logic_vector(15,8); -- 81
      when x"52"  => address_shifter <= conv_std_logic_vector(16,8); -- 82
      when x"53"  => address_shifter <= conv_std_logic_vector(17,8); -- 83
      when x"54"  => address_shifter <= conv_std_logic_vector(18,8); -- 84
      when x"55"  => address_shifter <= conv_std_logic_vector(19,8); -- 85
      when x"56"  => address_shifter <= conv_std_logic_vector(10,8); -- 86
      when x"57"  => address_shifter <= conv_std_logic_vector(11,8); -- 87
      when x"58"  => address_shifter <= conv_std_logic_vector(12,8); -- 88
      when x"59"  => address_shifter <= conv_std_logic_vector(13,8); -- 89
      when x"5A"  => address_shifter <= conv_std_logic_vector(14,8); -- 90
      when x"5B"  => address_shifter <= conv_std_logic_vector(5,8);  -- 91
      when x"5C"  => address_shifter <= conv_std_logic_vector(6,8);  -- 92
      when x"5D"  => address_shifter <= conv_std_logic_vector(7,8);  -- 93
      when x"5E"  => address_shifter <= conv_std_logic_vector(8,8);  -- 94
      when x"5F"  => address_shifter <= conv_std_logic_vector(9,8);  -- 95
      when x"60"  => address_shifter <= conv_std_logic_vector(0,8);  -- 96
      when x"61"  => address_shifter <= conv_std_logic_vector(1,8);  -- 97
      when x"62"  => address_shifter <= conv_std_logic_vector(2,8);  -- 98
      when x"63"  => address_shifter <= conv_std_logic_vector(3,8);  -- 99
      when x"64"  => address_shifter <= conv_std_logic_vector(4,8);  --100
      when others => address_shifter <= conv_std_logic_vector(50,8);
    end case;

    case address_shifter(7 downto 3) is
      when "00000" => CS1 <= "1111110";
      when "00001" => CS1 <= "1111101";
      when "00010" => CS1 <= "1111011";
      when "00011" => CS1 <= "1110111";
      when "00100" => CS1 <= "1101111";
      when "00101" => CS1 <= "1011111";
      when "00110" => CS1 <= "0111111";
      when others  => CS1 <= "1111111";
    end case;

    if    nMODE ='1' and nFIRST ='0' and address_send > x"32" then
      address_valid <= '0';
    elsif nMODE ='1' and nFIRST ='1' and address_send < x"33" then
      address_valid <= '0';
    else
      address_valid <= '1';
    end if;

  end if;
end process;

-- DAC MODULE OUTPUTS GENERATION

state_machine_process : process(nRESET,CLOCK)
begin
  if nRESET='0' then
    STATE       <= IDDLE;
    ENABLE_SYNC  <= '0';
    COMP_ENABLE  <= '0';
    COMP_DISABLE <= '0';
    DATA_INT     <= '0';
    reg_data     <= ( others => '0');
    count_seq    <= ( others => '0');
  elsif rising_edge(CLOCK) then
    reg_data <= reg_data(6 downto 0) & DATA ;

    case STATE is
      when IDDLE     => if ( DATAEN = '0' and ADDREN ='1' ) then
                          STATE     <= VERIF;
                          count_seq <= ( others => '0');
                        end if;

      when VERIF     => if address_valid = '1' then
                          STATE <= COMMAND;
                        else
                          STATE <= FINISH;
                        end if;

      when COMMAND   => if count_seq = "0101" then
                          STATE <= CMD_ACK;
                        else
                          if ADDREN ='0' then
                            STATE <= FINISH;
                          else
                            STATE <= COMMAND;
                          end if;
                          count_seq <= count_seq + 1;
                        end if;

      when CMD_ACK   => case reg_data is
                          when x"0A"  => STATE <= DAC_VAL0; -- DAC Valid
                          when x"05"  => STATE <= DAC_INH0; -- DAC inhibit
                          when x"AA"  => STATE <= DAC_DAT0; -- DAC Value
                          when others => STATE <= FINISH;
                        end case;

      when DAC_VAL0  => STATE       <= DAC_VAL1;
                        COMP_ENABLE <= '1';

      when DAC_VAL1  => STATE       <= FINISH;
                        COMP_ENABLE <= '0';

      when DAC_INH0  => STATE        <= DAC_INH1;
                        COMP_DISABLE <= '1';

      when DAC_INH1  => STATE        <= FINISH;
                        COMP_DISABLE <= '0';

      when DAC_DAT0  => if ADDREN ='0' then
                          STATE <= FINISH ;
                        else
                          STATE <= DAC_DAT1;
                        end if;

      when DAC_DAT1  => STATE       <= DAC_DAT2; -- send C/D bit to DAC
                        ENABLE_SYNC <= '1';

      when DAC_DAT2  => STATE    <= DAC_DAT3; -- send A2 bit to DAC
                        DATA_INT <= address_shifter(2);

      when DAC_DAT3  => STATE    <= DAC_DAT4; -- send A1 bit to DAC
                        DATA_INT <= address_shifter(1);

      when DAC_DAT4  => STATE    <= DAC_DAT5; -- send A0 bit to DAC
                        DATA_INT <= address_shifter(0);
```

```
                        count_seq <= ( others => '0');

      when DAC_DAT5  => DATA_INT <= reg_data(1);      -- send Datas to DAC
                        if count_seq ="1011" then
                          STATE <= DAC_DAT6;
                        else
                          if ADDREN ='0' then
                            STATE <= DAC_DAT6 ;
                          else
                            STATE <= DAC_DAT5;
                          end if;
                          count_seq <= count_seq + 1 ;
                        end if;

      when DAC_DAT6  => STATE       <= FINISH;
                        DATA_INT    <= '0';
                        ENABLE_SYNC <= '0';

      when FINISH    => if ADDREN ='0' then
                          STATE <= IDDLE;
                        end if;
    end case;
  end if;
end process state_machine_process;

-- VALIDATION COMMAND LINES FROM 1 TO 50 SETTINGS

CSVALID_PROCESS : for i in 0 to 49 generate
  CSVALID(i) <='1' when (address_shifter = conv_std_logic_vector(i,8)) else
               '0';
end generate CSVALID_PROCESS;

-- VALIDATION OUTPUTS GENERATION

VALID_PROCESS : for i in 0 to 49 generate
  process(nRESET,CLOCK)
    begin
      if nRESET = '0' then                          -- ON ATTEND LE RESET
        VALID(i) <= '0';                            -- VALID1 RESTE A ZERO
      elsif rising_edge(CLOCK) then        -- ON ATTEND UN FRONT D'HORLOGE
        if (COMP_ENABLE = '1' and CSVALID(i) = '1') then     -- ON TESTE
L'ENABLE ET L'ADDRESS
          VALID(i) <= '1';                     -- AFFIRMATIF => VALID = 1
        elsif (COMP_DISABLE = '1' and CSVALID(i) = '1') then – NEGATIF =>
VALID = 0
          VALID(i) <= '0';
        end if ;
      end if ;
  end process ;
end generate VALID_PROCESS;

-- OUTPUT SIGNAL FROM ACTEL CPLD TO DAC COMPONENT

process(nRESET,CLOCK)
begin
  if nRESET = '0' then
    nCS      <= ( others => '1');
    DATA_OUT <= ( others => '0');
  elsif rising_edge(CLOCK) then
    DATA_OUT <= DATA_INT & DATA_INT & DATA_INT
                & DATA_INT & DATA_INT & DATA_INT & DATA_INT ;
    if ENABLE_SYNC = '1' then
      nCS <= CS1 ;
    else
      nCS <= "1111111";
    end if;
 end if;
end process;

test(1) <= CLOCK ;
test(2) <= nRESET ;
test(3) <= DATA ;
test(4) <= ADDREN ;
test(5) <= DATAEN ;
test(6) <= nMODE ;
test(7) <= nFIRST ;
test(8) <= '1' when STATE = FINISH else '0';
test(9) <= '1' when STATE = iddle else
           '1' when STATE = VERIF else
           '1' when STATE = COMMAND else
           '1' when STATE = CMD_ACK else
           '0' ;

end RTL ;
```

Annex 13 : Documentation modifications

Version 1 (August 2005) : Initial version

Version 2 (May 2006) :

Update Board synoptic	Update
Add Annex 7 : Board Scheme	Add
Add Annex 8 : Board	Add
Add Annex 10 : Board implantation components (bottom side)	Add
Add Annex 12 : FPGA VHDL file description	Add
Add Annex 11 : High voltage command board connector designation	Add

Version 3 (August 2007) :

Annex 12 : FPGA VHDL file description	Update
Annex 8 : Board	Add
Figure 1 : Power connector pin assignment	Add
Chapter □o□ output connector to the HT module	Add
Chapter □o□ output connector to the HOUSEKEEPING board	Add
Chapter □o Power	Add
Chapter □o Amplifier and line level adapter & driver	Add
Annex 2 : HT module numeration for command and for housekeeping	Add
Annex 3 : FPGA Pin report (list per name)	Add
Annex 4 : FPGA Pin report (list per pin)	Add
Chapter 5 : Setting	Add
Annex 5 : FPGA signal generation	Add

TABLE OF CONTENTS

1. Overview 1
2. High voltage generators characteristics 1
2.1. Logical module characteristics 1
2.2. Analog module characteristics 1
3. In command serial link description 1
3.1. CHERCAM high voltage High level protocol 1
3.1.1. High Voltage level setting command 2
3.1.2. High Voltage disable/enable setting command 2
3.2. Serial link timing 2
4. DAC Board design 3
4.1. Board synoptic 3
4.2. input / output connector and level translator 3
4.2.1. input power connector 3
4.2.2. input serial link connector 4
4.2.3. output connector to the HT module 4
4.2.4. output connector to the HOUSEKEEPING board 5
4.3. Power 6
4.4. DACs 7
4.5. Amplifier and line level adapter & driver 7
4.6. FPGA 7
4.7. FGPA programmation connector 8
5. Setting 8
6. The Digital to Analog Converter : AD5328 (Analog Devices) 10
6.1. General description 10
6.2. Functional Block Diagram 10
6.3. Pin configuration and outline dimensions 10
6.4. Pin function descriptions 10
6.5. Power on reset 11
6.6. Input Shift Register 12
6.7. DAC Write 12
6.8. Timing characteristics 12

Annex 1 : Board synoptic 14
Annex 2 : HT module numeration for command and for housekeeping 15
Annex 3 : FPGA Pin report (list per name) 16
Annex 4 : FPGA Pin report (list per pin) 17
Annex 5 : FPGA signal generation 18
Annex 6 : FPGA Power consumption 19
Annex 7 : Board Scheme 21
Annex 8 : Board picture 39
Annex 9 : Board implantation components (top side) 40
Annex 10 : Board implantation components (bottom side) 41
Annex 11 : High voltage command board connector designation 42
Annex 12 : FPGA VHDL file description 43
Annex 13 : Documentation modifications 47

Figure 1 : Power connector pin assignment 4
Figure 2 : Serial link connector pin assignment 4
Figure 3 : Output connector to the HT module pin assignment (picture) 4
Figure 4 : Output connector to the HT module pin assignment (text) 5
Figure 5 : output connector to the HOUSEKEEPING board (pinout) 6
Figure 6 : components power supply 6
Figure 7 : Amplifier and line level adapter functional block diagram 7
Figure 8 : ProASIC PLUS ISP Board Layout and Programming Connector Top View 8
Figure 9 : DAC_BOARD functioning set mode 9
Figure 10 : AD5328 input shift register contents 12
Figure 11 : Serial interface timing diagram 13
Figure 12 : HT module numeration for command 15
Figure 13 : HT module numeration for housekeeping 15